Phylogenetic Reconstruction with Insertions and Deletions
[Full Version]

Alexandr Andoni* Mark Braverman† Princeton University Avinatan Hassidim‡ Bar-Ilan University

October 19, 2014

Abstract

We study phylogenetic reconstruction of evolutionary trees, undergoing three possible types of mutations: substitutions, insertions, and deletions. We give the first efficient algorithm for this problem which needs sequences of only poly-logarithmic length. The best previously known result of Daskalakis and Roch [DR10] showed how to do phylogenetic reconstruction using sequences of polynomial length. For the related problem of trace reconstruction (on an *a-priori* given tree structure), Daskalakis, Roch, and two of the authors [ADHR10] developed an algorithm that also uses only poly-logarithmic length sequences. The algorithm of [ADHR10], however, does not lead to phylogenetic reconstruction.

We develop three new tools enabling us to accomplish phylogenetic reconstruction under insertion/deletion mutations:

1. A new reconstruction guarantee for internal node sequences. The new guarantee is weaker but is more robust, allowing us to establish the reconstruction guarantee inductively.
2. A new local alignment and trace reconstruction algorithm, with an inductive correctness guarantee.
3. A new distance measure, tailored to traces estimated in the presence of insertions and deletions.

We analyze the error of the new algorithm by showing that it is dominated by a certain new random process on the tree, and apply percolation theory tools to analyze its behavior.

*andoni@mit.edu. The author is currently unaffiliated. Work done in part while the author was at Microsoft Research Silicon Valley, and previously at Princeton U./CCI, supported by NSF CCF 0832797.

†mbraverm@cs.princeton.edu. Work done in part while at Microsoft Research and Toronto University.

‡avinatanh@gmail.com. Work done in part while the author was at MIT and visiting Microsoft Research.

1 Introduction

The evolutionary history of a given set of species is usually modeled as a *phylogenetic tree*. The leaves of the tree correspond to the species which exist today. The root of the tree is their closest common ancestor, and each branching corresponds to a specification event, in which one species is extinct, and several new species are formed. The goal of *phylogenetic reconstruction* is to infer the tree structure from the information we have on the leaves, usually the DNA sequences of the species which exist today. The problem of phylogenetic reconstruction received a lot of attention in the recent literature; see, e.g., [Mos03, Mos04a, DMR06, Roc08, BRZ95, EKPS00, Iof96, BKMP05, MSW04, BCMR06, MHR08] and the excellent surveys [Roc07, SS03].

The basic model in the present work is the following common formalization. The phylogenetic tree is fixed arbitrarily (in particular, the tree need not to be balanced). The genetic information is modeled as a binary string (the CFN model of [Cav78, Far73, Ney71]). For each node, we call the bit string corresponding to the node its *trace*, and each location in this bit string a *site*. The genetic information of the root is assumed to be sampled uniformly at random from $\{0,1\}^k$ for some k representing the amount of the available genetic information. The reconstruction problem becomes easier as k increases. In every branching event, the bit string of the father node v is copied to each one of the child nodes $u_1, \ldots, u_d$, subject to a random mutation process. The mutation process of an edge (v, u_j) is characterized by three parameters: the substitution probability of the edge $p_s(v, u_j)$, the insertion probability $p_i(v, u_j)$ and the deletion probability $p_d(v, u_j)$. Given these probabilities, when copying the genetic information x_v to the child u_j, every site undergoes any of the following mutations, independently of any other site:

1. Substitution: the bit is flipped with probability $p_s(v, u_j)$.
2. Deletion: the bit is deleted with probability $p_d(v, u_j)$.
3. Insertion: a new site is added to the right of the current site with probability $p_i(v, u_j)$. The value of the bit in that site is a random bit in $\{0, 1\}$.

We also assume some bounds on the mutation probabilities. The substitution probability of every edge (v, u_j) is bounded from above and below: $\mathscr{P}_{\min} < p_s(v, u_j) < \mathscr{P}_{\text{subs}}$, where $\mathscr{P}_{\text{subs}}, \mathscr{P}_{\min}$ are global constants. The insertion and deletions probabilities are only bounded from above: $p_i(v, u_j), p_d(v, u_j) < \mathscr{P}_{\text{id}}$, for $\mathscr{P}_{\text{id}}$ subconstant. We call $\mathscr{P}_{\text{id}}$ the *indel* probability, and say that a site has undergone an indel if it suffered an insertion/deletion mutation.

We want to design an algorithm for the following phylogeny reconstruction problem. Consider an instance of the above evolutionary process on a tree with n leaves. Then, given an unordered list of leaves with their traces, and the parameters $d, \mathscr{P}_{\text{subs}}, \mathscr{P}_{\text{id}}, \mathscr{P}_{\min}$, the algorithm has to output the tree topology, identifying the tree leaves with the corresponding input traces. The algorithm is required to succeed for any choice of parameters $p_i(v, u_j), p_d(v, u_j), p_s(v, u_j)$ with high probability over the random coin flips of the evolutionary process (which determine the initial values of the root and govern the mutations). We limit our attention to algorithms which are polynomial in n, but make no further attempt to optimize the runtime.[1]

Traditionally, the computational biologists approach the problem via a two-stage process. Informally, first, they align all the sequences, and then they build the evolutionary tree on the aligned data. The advantage of this approach is that it allows one to build the tree assuming there are no indels. The success of this method is based on the assumption that insertions and deletions occur

[1] We note that, in exponential time, one can compute the maximum-likelihood estimate, reducing the problem to a purely information-theoretic problem.

much less frequently than substitutions. In practice, this method gives good results, although lately it was criticized in the biological community (see e.g. [WSH08, LRN+09a, LG08]).

In the theoretical community, most of the work focused on the second stage of building the tree, assuming that the genetic material is aligned correctly, and thus the reconstruction problem is reduced to one without indels. Under this assumption, early results (see, e.g., the work of Erdös et al. [ESSW99a, ESSW99b]) showed how to use distance estimations between all pairs of nodes in order to build the tree. Classically, these approaches require a polynomial number of sites, since the correlation between the same sites in two nodes which lie in different parts of the tree can be as small as $n^{-O(1)}$. More recently, the surprising result of Roch [Roc08] showed that reconstruction from distance information is possible even when k is subpolynomial.

In the last decade, several breakthrough results [Mos03, Mos04a, DMR06, Roc08] showed that there exists a critical substitution probability $\mathscr{P}^{\star}_{\text{subs}} = \frac{1}{2} - \frac{1}{2\sqrt{d}}$, such that if $\mathscr{P}_{\text{subs}} < \mathscr{P}^{\star}_{\text{subs}}$ the correct tree can be reconstructed with high probability even if the trace length is logarithmic, $k = O(\log n)$. A key, new ingredient in these results has been to reconstruct (approximations to) the traces of internal nodes of the tree iteratively. The authors of [Mos03, Mos04a, DMR06, Roc08] developed powerful frameworks for such reconstruction, where the requirement is not that the reconstruction is very good (in fact each bit is correct with i.i.d probability which is very close to 1/2), but rather that the algorithm fails at reconstructing any node only with an exponentially small probability (that is exponentially small in the length of the sequence k, which gives a failure probability much less than $1/n$). In fact all these papers take a union bound on the tree and argue that they never fail to reconstruct any node[2].

In this paper, we present phylogenetic reconstruction algorithms for the complete evolutionary process that includes indels, for the sequences of polylogarithmic length (or higher). We handle insertion and deletion probabilities of up to $O(1/\log^2 n)$. Before describing our results, we give an overview of related work.

1.1 Related Work

We now discuss past work on phylogeny reconstruction in the presence of substitutions and indels.

Daskalakis and Roch [DR10] proposed an algorithm for phylogeny reconstruction that uses only pairwise distance information between the (input) traces of the leaves of the tree. In particular, their method does not reconstruct the traces of the tree's inner vertices. Their algorithm works under the following assumptions:

1. The number of sites required for successful reconstruction is polynomial in the number of leaves, $k = \text{poly}(n)$.

2. The maximal indel probability is inverse proportional to the depth of the tree.

The latter assumption leads to an inverse-polynomial bound on maximal indel probability of at most $1/n^{\epsilon}$ for unbalanced trees.

To relax the above assumptions, it is natural to approach the phylogeny problem via trace reconstruction of internal nodes, following the general iterative framework of Mossel [Mos98, Mos01, Mos03, Mos04b], which allows one to reconstruct the tree from logarithmically-long traces. In this framework, one reconstructs the tree iteratively, level by level, maintaining a partial forest in the process. First, one clusters the leaves into a number of small groups, each corresponding to a set

[2]One could naively hope to reconstruct most of the tree, even when the reconstruction algorithm has some non negligible failure probability (say $1/n^{\epsilon}$). It turns out that this is not the case – a failure in one node can change distance estimations in the next level, and wreck havoc on the entire reconstruction.

of siblings. Each group is then joined into a mini-tree (a subtree), and one reconstructs the trace of the root of the resulting mini-tree. At the next stage, the process is repeated using the traces of the roots of the newly-formed mini-trees. This approach has two challenges:

1. Controlling the error of the recursive reconstruction, which might degrade over time: as one iteratively reconstructs traces at each stage from the previously-reconstructed traces, the reconstruction errors may accumulate. Mossel's 1998 paper shows that when there are only substitutions this does not happen: the magnitude of noise stabilizes at an absolute "small" constant.

2. The success of the reconstruction algorithm depends critically on always correctly selecting the traces to form each single mini-tree. Each intermediate reconstruction has some failure probability, which must be made small in order to apply a union bound over the entire tree. Otherwise, if the failure probability is not low enough (say, of the order of $1/n^{1-\epsilon}$ instead of $\ll 1/n$), one cannot hope to reconstruct even "most" of the topology, since if even one single node is not reconstructed correctly, then the clustering can change completely, and prevent reconstruction of the rest of the tree.

For the classical substitution mutation process, the results of [Mos03, Mos04a, DMR06, MHR08] tie the problem of trace reconstruction (reconstructing the trace of the root of a tree with a *known* topology) to phylogenetic reconstruction, as long as the trace reconstruction has *exponentially small* failure probability.

In presence of indel mutations, the trace reconstruction problem on a tree has been recently studied in [ADHR10], who showed that a *polylogarithmic* number of sites is sufficient for good (approximate) trace reconstruction, when the tree is d-ary for $d = \Omega(1)$. Their reconstruction procedure has the advantage that it works even if one of the inputs is adversarial[3]. However, it has a failure probability of $\Omega(1/\log n)$, which is exponentially worse than the one required for phylogenetic reconstruction in the Mossel framework.

On a more technical level, the [ADHR10] algorithm uses anchors, which are bitstrings of length $O(\log n)$, and islands, which are long sequences between the anchors. A node could be reconstructed only if:

1. Generating its children, there were no indels in any of the anchors.

2. No island suffered too many deletions.

Given that both conditions hold, one can do an alignment based on the anchors, and take a place-wise majority on the islands, which is the algorithm used in [ADHR10]. In essence, the approach manages to circumvent dealing with the indels, by not reconstructing the parts of the tree where they actually happen.

It is tempting to try to use the [ADHR10] approach to reconstruct at least a part of the topology, where not many indels occurred. However, this seems challenging: the parts which were not reconstructed correctly may well interfere with the clustering, and prevent reconstruction even of the parts of the tree in which no indels occurred. Nonetheless, some insights of the [ADHR10] algorithm form the departing point for our algorithms for phylogeny reconstruction.

On the empirical side, there has been a large body of work performing alignment when there are indels (see e.g. [TKF91, TKF92, Met03, MLH04, SR06, RE08] and the textbook [Fel04]). In

[3]This assumption requires [ADHR10] to work with large d, and limits extending their results to $d = 2$, which is the biologically interesting case.

a breakthrough paper, Wong et al. [WSH08] showed that factoring away indels and then inferring the phylogenetic tree assuming only substitutions does not work well, even in simple cases (e.g. even for a small tree which consists of seven species of yeast). In follow up work, Loytynoja and Goldman [LG08] presented heuristics for reconstruction which are based on alternating between performing alignment (to identify indels and factor them out), and building an evolutionary tree, assuming that there are only substitutions. An improved heuristic was presented by [LRN$^+$09b].

1.2 Our Results

In this work we give phylogenetic reconstruction algorithms in presence of insertion and deletion mutations for sequences of length $O(\log^2 n)$. The algorithms can handle insertion and deletion probabilities as high as $\Omega(1/\log^2 n)$. The results are obtained for both binary and d-ary trees, and for complete and general trees. We start the presentation on phylogenetic reconstruction of complete balanced d-ary trees (for large but constant d), which is the simplest case. We then present the additional tools required for the binary trees and the general (unbalanced) trees. Our main theorem reads as follows, and is proven in Section 4.2.

Theorem 1.1 *For every probability $\mathscr{P}_{\min} > 0$, there exist a (small) constant $\epsilon > 0$, and (large) constants C_1, C_2, C_3 such that, if $\mathscr{P}_{\text{id}} < \epsilon/\log^2 n$, the degree $d > C_1$, the substitution probability is at most $\mathscr{P}_{\text{subs}} \leq \frac{1}{2} - \frac{C_2\sqrt{\log d}}{\sqrt{d}}$ and the number of sites is $k > C_3 \log^2 n$, then the following holds. Fix a complete balanced d-ary tree and an indel process on it. Then, given the sequences of the n leaves of the tree, one can reconstruct the structure of the tree with high probability.*

When there are no indel mutations, the highest possible value for $\mathscr{P}_{\text{subs}}$ is $\frac{1}{2} - \frac{1}{2\sqrt{d}}$ [DMR06, Mos03], for any $d \geq 2$. As our analysis does not attain this threshold, we did not try to optimize C_1, C_2, C_3 and ϵ.

We note that for indel probability $\mathscr{P}_{\text{id}} \geq \Omega(\log\log n/\log n)$, the reconstruction is impossible if the length k of the root sequence is polylogarithmic in the number of leaves n. In such a setting, some of the leaves will be empty (all sites are deleted). Our algorithm works for $\mathscr{P}_{\text{id}} = \Theta(1/\log^2 n)$.

We also show that our results extend to binary trees. For balanced binary trees, we prove the following theorem in Section 5.

Theorem 1.2 *For every (small) constant δ there exist a (small) constant ϵ, and a (large) constant C_3 such that, if the indel probability $\mathscr{P}_{\text{id}} < \epsilon/\log^2 n$, the maximal substitution probability $\mathscr{P}_{\text{subs}} < \epsilon$, the minimum substitution probability $\mathscr{P}_{\min} = \delta \cdot \mathscr{P}_{\text{subs}}$ and the number of sites in the root is $k > C_3 \log^2 n$, then given the sequences of the n leaves of the tree, one can reconstruct the structure of the tree with high probability.*

We also give an algorithm for general binary trees. Before stating this result, we simplify the mutation process (for reason which will be explained bellow). Every edge e has a length $l_e \in [\delta, 1]$. The mutation process changes to be as follows for an edge e:

1. Each bit gets flipped with probability $l_e \cdot \mathscr{P}_{\text{subs}}$.
2. Let s be a sample from a random geometric variable with expectation $k \cdot l_e \cdot \mathscr{P}_{\text{id}}$. We perform s insertions and s deletions in random places in the string.

Under this model, we prove the following theorem for general binary trees in Section 6.

Theorem 1.3 *For every $\delta > 0$ there exists a (small) constant ϵ, and a (large) constant C_3 such that, if $\mathscr{P}_{\text{id}} < \epsilon/\log^2 n$, $\mathscr{P}_{\text{subs}} < \epsilon$, $\mathscr{P}_{\min} = \delta\,\mathscr{P}_{\text{subs}}$, and the number of sites in the root is $k > C_3 \log^2 n$, then given the sequences of the n leaves of the tree, one can reconstruct the structure of the tree with high probability.*

We now motivate the modification in our model. The changed model is equivalent to previous one, conditioned on the fact that the length of the sequence does not change under mutation. This prevents us from obtaining empty leaves, and leaves us with a reversible evolutionary model (when discussing general trees it is common to assume that there is no root). We call this model INDEL-REV$_\delta$ to emphasize that it is reversible.

Without conditioning on the length, depending on the topology, one may have empty leaves with high probability even if we enforce (say) that the deletion probability is equal to the insertion probability. Requiring that each edge has a well defined length was done to simplify the analysis. The results would also hold if instead we were given a minimal indel probability.

2 Preliminaries

Notation. We fix some standard notation. We use $[d]$ to denote the set $\{1, \ldots, d\}$. When we say that an event happens with high probability, we mean with probability at least $1 - 1/n^c$, where we can set up the parameters such that c is as big as we need.

Given a bit string x, we let $x[i : j]$ denote the $j - i$ bits from location i to location $j - 1$ in x. We let $|x|$ denote the number of 1's in x. Sometimes we wish to distinguish between a node v and the sequence of bits it has; in this case, we usually denote the sequence by x_v.

Sequence distances. Given two strings x, y, we define the *agreement* $\mathscr{A}(x, y, \gamma)$ to be

$$\mathscr{A}(x, y, \gamma) = k - \min_{x' \in \text{ed}_\gamma(x)} |x' \oplus y|$$

where $\text{ed}_\gamma(x)$ is the set of sequences obtained from x by performing up to γ insertions/deletions operations and k is the length of x. If $\mathscr{A}(x, y, \gamma) \geq 0.75k$, we define

$$\text{d}_{\text{ed}}(x, y, \gamma) = -\log\left(\frac{2\,\mathscr{A}(x, y, \gamma)}{k} - 1\right).$$

If $\mathscr{A}(x, y, \gamma) < 0.75k$, we say that $\text{d}_{\text{ed}}(x, y, \gamma)$ is not well defined (the algorithm only uses short distances).

When there are no insertions and deletions, the most common distance measure used is the logarithm of twice the average correlation minus 1. Our distance reduces to the classical distance, by allowing zero edit operations. In the substitution only case, this distance measure is (approximately) additive, when two vertices are close by [MHR08]. For example, if u_1 is the father of u_2 who is the father of u_3, then

$$\text{d}_{\text{ed}}(u_1, u_2, 0) + \text{d}_{\text{ed}}(u_2, u_3, 0) \approx \text{d}_{\text{ed}}(u_1, u_3, 0)$$

Alignments. For each node v, we define d functions $f_i^v : \{1, \ldots K_v\} \mapsto \{1, \ldots K_{v(i)}\} \cup \{\perp\}$, where K_v is the length of the sequence at node v, $v(i)$ is the i'th child, and $K_{v(i)}$ is the length of the i'th child. We let $f_i^v(j)$ denote the place in v_i which the j'th site went to, or $\perp$ if the site was deleted. Thus, each f_i^v is strictly monotone. When v is clear from context, we omit it.

During the algorithm we will reconstruct the phylogenetic tree recursively, level by level. It will be convenient to distinguish between the *ideal tree*, which is the tree generated by the random process, and the *reconstruction tree*, which is the tree reconstructed by the algorithm. Variables which refer to sequences of bits in the reconstructed tree will have a hat. There will also be a clear correspondence between nodes in the ideal tree and the ones in the reconstructed tree. Thus, if x_v is the subsequence of bits in node v in the ideal tree, $\hat{x}_v$ is the sequence of bits in the node which corresponds to v in the reconstructed tree.

Problem parameters. We will usually think of each x_v or $\hat{x}_v$ as composed of *blocks*, which are consecutive sequences of length B, where $B = O(\log n)$. The algorithm is only going to use the first $O(\log n)$ blocks. To simplify the notation, the paper is written so that the algorithm uses B blocks, so it is enough to remember that B is a large constant times $\log n$ (the number of blocks does not have to be equal to the length of each block in order for the algorithm to succeed). We will also have *bad* blocks (which will also be called *red blocks*), and we will later prove that with high probability every vertex has at most α red blocks, where $\alpha = O(\log n) \ll B$. In the balanced tree, we also prove that with high probability on the path from the leaf to the root there are at most α indel operations. The use of the same bound for both these variables is again done to enable the reader to remember fewer parameters. The reconstruction guarantee for d-ary trees will introduce a new constant β, and it will reconstruct most of the vertex, such that each bit is correct with probability $1 - \beta$. We can use $\beta = O(d^{-2/3})$ a small constant. As we explain below, in the binary case this guarantee is not enough, and we present a new reconstruction procedure which reconstructs each "good" bit correctly with probability at least $1 - \beta$ and at most $1 - \beta + \gamma$.

3 Overview of Techniques

We begin by giving a high level overview of the algorithm for complete d-ary trees for large d and the technical tools involved. We then show how to extend these tools to the case of binary (balanced) trees and general trees, and explain what new tools we introduce. In this section we focus on intuition, skipping the details; in later sections we give formal proofs of our theorems.

For complete trees, our algorithm follows the classic version of the iterative approach given by Mossel [Mos98, Mos01, Mos03, Mos04b]. The general form of the algorithm appears in Alg. 1.

Algorithm 1: General structure of the phylogeny reconstruction algorithm.

1 **for** $j = 0$ *to* $\log_d n$ **do**
2 The algorithm begins with all the vertices of level j (for $j = 0$, these are the input sequences representing the leaves);
3 Partition the nodes of level j into clusters of siblings. Each set of siblings will contain d vertices;
4 From each cluster of d siblings, reconstruct the trace of the root of the cluster (the father) which is a node at height $j + 1$;

There are two major components to describe: 1) how to partition the nodes into clusters of siblings (step 3 of the algorithm), and 2) how to perform the trace reconstruction of the root of the cluster, which can be used recursively (step 4). We start by describing how we approach the second component, the trace reconstruction.

In the classical substitutions-only case, Mossel's algorithm reconstructs the father of a cluster using recursive majority. When there are indels, we need to perform some alignment of the strings

in a cluster. As the indel probability is fairly low, most of the time, this alignment will not be necessary, since indels will not occur in the relevant sequences[4]. On the other hand, when indels do occur, alignment becomes challenging as it is hard to identify the "same sites" in the cluster strings in order to apply a majority on them (for example when you have several indels which are close to one another it is very hard to identify them). Small misalignment events also introduce non-trivial correlations between reconstructions of different traces, which may lead to an uncontrolled accumulation of noise/error and adversly affect the reconstruction down the road. Thus, it is important to localize the errors by basing the decisions of the algorithm on high probability events. This motivates doing the alignment in blocks of length $B = O(\log n)$, where B is a large constant. We note that [ADHR10] also use $O(\log n)$ substring for alignment, but their algorithm is based on "anchors", which were assumed to have no indels, or else the entire reconstruction procedure for the cluster fails. Our algorithm depart from this idea and uses a new reconstruction guarantee, described below.

Our local alignment is presented in Alg. 2.

Algorithm 2: Trace reconstruction of a node v from the traces of its d children.

1 Partition each child into blocks of length B;
2 Pick a special child s;
3 Let $A_{b,s} = \hat{x}_s[b \cdot B : b \cdot B + B]$ be the b'th block of the special child;
4 **for** *every child* $t \neq s$ **do**
5 Find a consecutive substring $A_{b,t}$ of $\hat{x}_t$ of length B which "matches" block $A_{b,s}$ of $\hat{x}_s$. Look for such substrings only in blocks $b-1, b, b+1$ of $\hat{x}_t$;
6 The b'th block of the father is the majority vote over all t of $A_{b,t}$;

We now sketch the recursive guarantee (inductive hypothesis), which lies at the heart of the analysis. Let x be the original string, and $\hat{x}$ be the reconstructed string. We require that there exists a function g that aligns between $\hat{x}$ and x. The alignment is subject to some constraints. Divide $\hat{x}$ into blocks of length B, each labeled either *red* or *green*, and each red block R has a number $n_R \geq 0$ associated with it. We require that g is constant over green blocks, and that it can change by at most n_R in each red block. That is, if $j, j+1$ are sites in a green block then $g(j) = g(j+1)$, and if i, j are sites in different red blocks then $|g(i) - g(j)| \leq \sum n_R$ where the sum ranges on all the red blocks between i and j. Let y be the string which is the aligned version of $\hat{x}$, that is $y[j] = \hat{x}[j + g(j)]$. If j is in a red block, we have no guarantee on $y[j] \oplus x[j]$. Let $z[j] = x[j] \oplus y[j]$, and consider only the places where j is in a green block. The vector z in those places is stochastically dominated by a random variable which has 1 in each position with i.i.d. probability β, for some small β. In particular, this means that if j is green

$$\Pr(\hat{x}[j + g(j)] = x[j]) \geq 1 - \beta$$

but the probability may be bigger than $1 - \beta$, which actually complicates things, as we discuss below. The guarantee is formally defined in Section 4.2.

To better understand the recursive guarantee, we illustrate how a "typical" reconstruction could look like. Let $u_1, \ldots u_d$ be d leaves who are siblings, and let x denote the father sequence which generated them. The reconstruction of the leaves is trivial (just the inputs themselves), and all

[4]This is the approach presented in [ADHR10]. They show that since most of the tree is easy, they can probably reconstruct most of the tree (given the topology). For example, they show that when there are $\log^2 n$ sites and with an indel probability which is $o(1/\log^2 n)$ ignoring the indels altogether and just doing recursive majority reconstructs the root (again assuming the topology is given).

the blocks are green. As we show in the distance estimation part, if most of the blocks are green, we can measure distances with high enough accuracy, and all the leaves are identified as siblings. Letting $\hat{x}$ denote the reconstructed version of the father, we try to characterize the relation between x and $\hat{x}$ given the indel structure of this part of the tree. Assume without loss of generality that the special child is u_1:

1. The reconstructed bit string of the father $\hat{x}$ is aligned according to u_1. Suppose that the first bit is deleted going from the father to u_1. In this case, going from $\hat{x}$ to x will require an insertion right at the start, and the first block of $\hat{x}$ will be red. More generally, in the worst case, every indel going from x to u_1 may require an indel operation to go from $\hat{x}$ to x. Thus whenever there is an indel going from the father to u_1 we will have a red block. Since the local alignment we use only shifts blocks by at most 1, even if there are several indels happening very close to each other, each will get its own red block. To remember this, we give the first block the number of indels (so this block will have a non zero number).

2. After the local alignment, we take a placewise majority, between all the children. Suppose that there were no indels going from the father to u_1, but there were deletions in the first block, going from the father to some of the children. Since the local alignment is done in blocks, we will compute the majority of descendants of different sites in the father. This introduces a dangerous kind of noise in computing the father: noise which has correlations. Instead of controlling these correlations, if two or more children have indels in the same block, we give in on the corresponding block at the father.

Suppose again that the number of red blocks at the parent level is "small". We now use the distances to cluster the parents, and continue to the grandparents. Now the nodes which were reconstructed at the previous level are the children. However, since we use the reconstructed version, these nodes themselves have red blocks, and we need to describe what happens in the reconstruction when the children have red blocks (with numbers), and there are indels:

1. If there are indel operations going from the father to the special child, or the corresponding block in the special child is red, the father's block will be red, and the number on it will be the number of the block of the special child (zero if the corresponding block in the special child was green) plus the number of indels going from the parent to the special child.

2. Consider the b'th block of the parent. If at least two children have red blocks at block b, or have indels going from the parent to them at block b, we say the parent has a red block in block b. However, consider the case where (say) two children have red blocks in block $b-1$. In this case, the local alignment procedure may move the end of these blocks to intersect with block b of the parent. Therefore if at least two children have indels (or red blocks) in blocks $b-1$, b or $b+1$ we declare the parent's block red.

The propagation of red blocks as presented here has two important properties:

1. It is essentially independent of the algorithm, and depends only on the indel structure (as long as no single node has too many red blocks - so the distance estimations are good enough)

2. It has a (limited) way of eliminating red blocks, and handling indels: if (locally) at most one child has a red block (or an indel), and this is not the special child, the indel will be corrected.

To finish the argument, we need to show two things:

1. If the algorithm errs, then the process we defined has a red block, so the process "dominates" the algorithm. This is done by closely following the algorithm, and seeing how it behaves on red blocks and on indels.

2. Show that the number of red blocks in the inductive process is never too large. Formally, we show that the number of red blocks will never be more than α, where α is a small multiple of $\log n$.

So far we gave intuition as to how the recursive guarantee propagates upwards, and why the algorithm is dominated by the process. Showing that (with high probability) the number of red blocks is not too larger requires more work. Ignoring the fact that neighboring blocks can effect one another, the standard way to do this, is to apply Galton Watson analysis, and show that given that the father is red in block b, the expected number of children for which block b is red is less than 1. Even without the effect of neighboring blocks, this fails, as given that block b of the father is red, the expected number children for which block b is red is *more* than 1. Thus, naive Galton Watson analysis would not yield any bound on the number of red blocks.

In order to get around this difficulty, and to handle the effect of neighboring blocks, we notice that this process is defined bottom up (unlike many other processes on trees which are defined top down). Thus, careful analysis reveals that the size of the red connected components is small, and this enables to bound the effect of neighboring red blocks, and the maximal number of red blocks in a single node of the tree. The analysis is the most involved part of the reconstruction, and is presented in Section 4.1 (with Lemma 4.2 giving the formal result).

We now turn to describing the other major component of the phylogeny reconstruction: partitioning the current level of the tree into clusters of siblings. The approach originated by Mossel requires this stage to be done without errors. Formally, this is justified by the requirement that the entire tree structure is reconstructed correctly. However, one could consider a relaxed version of phylogenetic reconstruction, in which we are trying to build the tree but allow a few mistakes[5] (in fact it is believed that the trees reconstructed by computational biologists sometimes have small mistakes, but they are nonetheless useful). The technical reason for this requirement is that approximate phylogenetic reconstruction does not seem any easier than perfect phylogenetic reconstruction. Any error in the clustering, would insert some erroneous data to the reconstruction procedure. These errors are unlikely to be random errors (for example the rouge vertex which is inserted to the cluster is not a random vertex but a vertex chosen by the clustering algorithm), and thus they are hard to control[6]. Thus, in all other algorithms any error in clustering is likely to propagate upwards all the way to the root.

Our clustering algorithm uses a new distance measure between (reconstructed) traces. The distance is a hybrid between Hamming distance and edit distance. We note that neither of the two classical distance would have been suitable by itself. While Hamming distance is obviously jeopardized by misalignments, to understand why edit distance fails we need to consider the use of the distance estimates in the reconstruction. The goal of the distance estimates is to perform the clustering, and find siblings. Unfortunately, since edges have different lengths (different mutation probabilities), the closest vertex to i might not be his sibling j, but another vertex. This is true even when there are no indels operations, and there is a classical way to get around this, called the Four Points Condition, or neighbor joining. Essentially, it says that if we pick a distance measure

[5] One has to be careful in defining what is an approximate tree. Building 0.99 of the tree can be done without any reconstruction or fancy technologies just by looking at the short distances and building the bottom part of the tree correctly. As before, it is the "hard" parts of the reconstruction which present the challenge.

[6] Failing to reconstruct $n/\operatorname{poly}\log n$ of the vertices is the reason [ADHR10] can not be used for phylogenetic reconstruction even in the balanced d-ary case

which is additive, then if i, j are siblings then for any other two vertices x, y we have

$$dist(i,j) + dist(x,y) < dist(i,x) + dist(j,y)$$

When there are no indels, the additive variant of Hamming distance being used is $\mathrm{d}_{\mathrm{ed}}(x, y, 0)$, or minus log of the average correlation between x and y. Edit distance does not satisfy the required additivity conditions.

We now mention another challenge in estimating distances. The reconstruction guarantee says that in the green blocks we are correct with probability *at least* $1-\beta$, and not *exactly* $1-\beta$ (which is the case when there are no indels). The guarantee we have results in an additive error of β, which could be adversarial. To see why this guarantee is so weak, consider two far away nodes x_u, and x_v, which are completely uncorrelated. If $\hat{x}_u$ is a reconstruction of x_u such that each bit is correct with probability $1-\beta$, and $\hat{x}_v$ is a reconstruction of x_v such that each bit is correct with probability $1-\beta$, then $\hat{x}_u$ and $\hat{x}_v$ are uncorrelated for any value of β. However, if $\hat{x}_u$ ($\hat{x}_v$) is a reconstruction of x_u which is correct at every bit with probability at least $1-\beta$, the situation can be very different. Suppose that whenever $x_u[i] = x_v[i]$ agree, then $\hat{x}_u[i] = x_u[i]$ and $\hat{x}_v[i] = x_v[i]$, but whenever $x_u[i] \neq x_v[i]$, then $\Pr(\hat{x}_u[i] = x_u[i]) = 1-\beta$, and also $\Pr(\hat{x}_v[i] = x_v[i]) = 1-\beta$. In this case, $\hat{x}_v$ and $\hat{x}_u$ will have a correlation of $2\beta - \beta^2$. This can of course create trouble, especially if β is large. To handle this in the d-ary case, we use d large enough such that β is smaller than $\mathscr{P}_{\min}$. In the binary case, this will require a different analysis.

The final property that we require is that the distance will be robust to edit distance errors introduced by the reconstruction. The distance we end up using is d_{ed}, which allows for a few "free" edit distance operations (to compensate for red blocks and local deletions[7]), but essentially behaves like Hamming distance with respect to concentration and additivity when looking at the logarithm. The properties of the distance are discussed in Section 4.4.

3.1 Binary trees

Similarly to the substitution only case, we reconstruct $\log d$ levels at a time, where $\log d$ is a suitably chosen constant.

The main difference between d-ary trees (with large constant d) and reconstructing $\log d$ levels at a time in a binary tree, is that the error on each bit in the green blocks can be large. In the d-ary case, we took d to be such that the error was less than $\mathscr{P}_{\min}$, and now it is greater than $\mathscr{P}_{\mathrm{subs}}$. This means that we can not afford an additive adversarial error of β in our distance approximation. We note that in the substitution only case, the error on each bit was close to a half, and indeed it was larger than the substitution probability. However, in the substitution only case the probability that $\hat{x}$ agreed with x was exactly $1-\beta$, and thus this error averaged out.

To eliminate the adversarial influence, we replace the place wise recursive majority with a threshold function. That is, we replace Step 6 in the algorithm sketch presented in Section 3 by counting the number of ones. If there are more than $2d/3$ ones we reconstruct the father as 1, if there are less than $1/3$ ones, we reconstruct the father as zero. Otherwise we flip a random coin to determine the value at the father. Suppose the adversary has complete control over a small *fraction* of the entrances to the threshold function, and no control over the rest of the entries. Then the adversary can tilt the outcome, only if the random entries are distributed near $d/3$, or over $2d/3$. Since $1/3$ has no finite binary representation, this event happens with low probability, which can be

[7]The hybrid distance can not compensate for indels which happened between two far away nodes, e.g. in the general tree case. It is still sufficient for our needs since we do not use distance information between far nodes.

bounded by $\mathscr{P}_{\text{subs}}{}^{100}$ for d large enough[8]. The exact analysis of the process is somewhat involved, and takes most of Section 5.5.

The rest of the work in doing the binary case is around the distance estimation, and the red and green trees.

3.2 General binary trees

All the tools we designed for the complete trees were built in a modular way, so that they can be used inside a scheme which reconstructs general binary trees when there are no indels. Still, proving that all the tools fit in place requires a large amount of work.

We use the reconstruction scheme of Mihaescu, Hill and Rao [MHR08], which is a simplification of [DMR06]. This scheme maintains a forest, and glues trees together, in a way which is very similar to the cluster approach introduced by Mossel. When the trees become tall enough, the inner vertices are being reconstructed, using the reconstruction scheme due to Mossel.

In order to obtain reconstruction of general trees we introduce a slight change in the indel process. We no longer assume that each edge has a probability for insertion and a probability for deletion (in this case we will have empty leaves even if we assume these probabilities are equal), but rather that the number of insertions is equal to the number of deletions on each edge. This is done to obtain a constant length of the genetic sequence, as well as reversibility.

In the general trees, we need to prove an analog of Lemma 4.1, which says that nodes which are close in tree metric will not suffer too many indels. The analog of this Lemma is used to show that the hybrid distance is well defined on nodes which are close in the tree metric. Since the indel probability is close to the substitution probability, these are the nodes we care about.

We also need to prove that the distance estimates are correct. This is done in a manner similar to the proof of Lemma 8.2, and takes into account both the adversarial errors and the random errors.

Then we need to show that the global process does not generate too many red blocks. The proof of this fact is based on the analysis of the process, which is based on local components and a union bound. As the number of leaves is still n, the same union bound apply.

Finally, we need to change the reconstruction guarantee, to fit the case where the reconstruction is made from an inbalanced tree. When there are no indels, this just results in using a weighted majority. Here we need to be a bit more careful, giving the children the correct probability to be the special child, and proving that threshold reconstruction still gives the correct guarantee.

The organization of the rest of the paper is as follows. Section 4.1 describes the coloring process in more detail, and proves that with high probability the number of red blocks, and the sum of numbers on red blocks is bounded. Section 4.2 introduces the algorithm and the reconstruction guarantee, and Section 4.3 proves that the algorithm is dominated by the percolation process. Section 4.4 shows how to use the reconstruction guarantee to find siblings. Finally, sections 8,9 give the proof for binary trees and for general trees.

4 Phylogenetic Reconstruction for High-Degree Trees

We now describe the algorithm and analysis for the problem of reconstructing the phylogenetic tree for a balanced tree of arity $d = \Omega(1)$, thereby proving Theorem 1.1. We will show the extension to balanced binary and general binary trees in subsequent sections.

[8]Note that the event that about half the entries agree occurs with probability about $2\,\mathscr{P}_{\text{subs}}$ - it's enough that there is a substitution in one of the top two branches. The analysis can be made such that the probability of this event is $\mathscr{P}_{\text{subs}}{}^{2}$.

The proof is composed of several parts. First, we introduce and analyze the random process that controls the error propagation along the tree during our reconstruction algorithm. Second, we present our trace reconstruction algorithm and the iterative guarantee. Third, we show that the algorithm reconstructs the traces of the nodes correctly, up to noise that is dominated (controlled) by above random process. Finally, we show that, given that the algorithm reconstruct the traces approximately correctly, the tree topology is reconstructed correctly.

4.1 Red and Green Trees

We now introduce and analyze the random process that will be shown to control the error of our algorithm. Here, we assume fixed (ideal) tree. Let B^2 denote the length of the sequence of the root. Most of the properties we prove are defined on blocks, which are consecutive sequences of sites of length B, which begin at kB and end at $kB + B$ for some value k. The following claim motivates this.

Claim 4.1 *With high probability, the maximum number of indels between the root and any leaf is bounded by $\alpha = O(\log n)$, and $\alpha \ll B$.*

Specifically, the condition from Claim 4.1 means that throughout the process indels never create shifts longer than α, which is much less than the length of a single block. The claim follows immediately from a standard concentration bound. We condition the rest of the analysis on the high probability event of Claim 4.1, without mentioning it explicitly. Given this event, we partition the sites of each node into B blocks, each of length B, except maybe the last block (which can be a bit longer or shorter). In the rest of the section, we ignore the length of the last block, implicitly assuming that it is B exactly[9].

We color all the blocks in the tree in two colors: red and green, and assign each red block some integer. Initially red nodes signify a misalignment between a parent node and a child node caused by indels. We then apply the following recursive process, level by level, from bottom to top.

Initialization: Assume that there are k an indel operation going from the father to the i'th child, which happen in the j'th block. Color block j red, and assign it the number k.

After coloring level $\ell - 1$, we color level ℓ by the Algorithm 3. This procedure temporarily expands the red blocks on level $\ell - 1$. These red blocks are only added to simplify the description (one can think of Algorithm 3 as first copying the child nodes, and then coloring the father while changing the temporary copies of the children). Algorithm 3 is carefully tailored to dominate misalignments in our reconstruction algorithm. By analyzing it we obtain the bounds that we need for the algorithm. For example, the temporary extension in the recursive coloring (line 9) corresponds to the fact that during the algorithmic reconstruction, errors that appear in two different children may "spill over" one block when we reconstruct the parent.

The following lemma is the heart of our error analysis, and its proof takes the rest of this section.

Lemma 4.2 *With high probability, in each node of the tree there are at most α blocks which are not green, assuming $\alpha \geq \frac{1200 \log d}{\log 1/(B^2 \cdot \mathscr{P}_{\mathrm{id}})}$.*

In the remainder of the paper, we will condition the rest of the analysis on the high probability event of Lemma 4.2, without mentioning it explicitly.

[9]Handling the length of the last block requires some tedious details, but is not fundamentally different. In fact, in the i'th level of the tree, we could throw away the last block. This would mean that in the top level the root would have $B - \log n$ blocks, which would still not affect the distances by much because B is a large constant times $\log n$.

Algorithm 3: Recursively coloring a father y given the d child nodes

1 **Initialization:**
2 **for** *each vertex* t*, and each block* k **do**
3 **if** *there are* $n_{k,t} > 0$ *insertions and deletions going from* t*'s parent to* t *in block* k **then**
4 Color block k in t red, and give it the number $n_{k,t} > 0$;
5 **Recursively coloring a father** y **given the** d **child nodes:**
6 Let s be a random child ;
7 **for** *every child* $t \neq s$ **do**
8 **for** *every maximal consecutive sequence of red blocks in the* t*'th child,* $i, i+1, \ldots, i+k$ **do**
9 temporarily color blocks $i-1, i+k+1$ red in the child t, and give them the number 1.
10 **for** $k = 1$ *to* B **do**
11 **if** *the* k*'th block in* s *is red, and has number* i **then**
12 Color the k'th block in the father red, and add number i to it.
13 **else**
14 **if** *exist distinct children* t_1, t_2*, in which the* k*'th block is red* **then**
15 Color the k'th block in the father red, and give it number 1.

Proof of Lemma 4.2
We distinguish between *initial* red blocks that were placed during the initiation phase (lines 2-4) and *acquired* red blocks that were passed from child to parent (lines 6-15).

We first note that the probability that a node contains *any* initial red blocks at all is bounded by ε_1 where $\varepsilon_1 < B^2 \cdot \mathscr{P}_{\text{id}}$ is small. We say that a node is red if it contains any red blocks (initial or acquired). We first claim that with high probability the largest connected component of red nodes is small.

Claim 4.3 *Except with probability* n^{-3}*, the largest red connected component in the graph has fewer than* $\varepsilon_2 \log n$ *nodes with initial red blocks, where* $\varepsilon_2 < \frac{3 \log d}{\log 1/\varepsilon_1}$ *is small.*

Proof: For a node v in the graph, denote by $P_i(v)$ the probability that the set S_v of initial red descendants of v that are connected to it through a red path contains at least i nodes. We will prove that

$$P_i(v) < \frac{\varepsilon_3^{i+1}}{i^2},$$

where $\varepsilon_3 = 40\sqrt{\varepsilon_1}$ is a small constant. We will show this by induction.
Base case: For S_v to be non-empty, one of the three cases has to hold: (1) v has an initial red block; (2) v has at least two red children; (3) v has one red child that has been randomly selected. This implies the following inequality:

$$P_1(v) \leq \varepsilon_1 + d^2 P_1(v)^2 + \frac{1}{d} P_1(v).$$

It is not hard to see that for a sufficiently small ε_1, $P_1(v) < 1/(2d)$, and thus we get that $P_1(v) < 3\varepsilon_1 < \varepsilon_3^2$.
Step: We want to show the bound for $P_i(v)$, $v > 1$. As in the base case, there are three possibilities that cover all the cases when $|S_v| \geq i$: (1) v has an initial red block; (2) v has at least two red

children; (3) v has one red child that has been randomly selected. Denote the probabilities of the three cases by Q_1, Q_2 and Q_3. Given that there is an initial red block in v, the probability that $|S_v| \geq i$ is bounded by the probability that it is $\geq i-1$ without this information. Thus $Q_1 \leq \varepsilon_1 \cdot P_{i-1}(v)$. We also have $Q_3 \leq \frac{1}{d} P_i(v)$. Thus we have

$$P_i(v) \leq \varepsilon_1 \cdot P_{i-1}(v) + Q_2 + \frac{1}{d} P_i(v) < \frac{1}{3} \frac{\varepsilon_3^{i+1}}{i^2} + Q_2 + \frac{1}{3} P_i(v).$$

To complete the proof, all we need to show is that $Q_2 < \frac{1}{3} \frac{\varepsilon_3^{i+1}}{i^2}$. To estimate Q_2 we cover it using the following events. For each $0 < j < i$ and index $1 \leq k < d$ let Q_{jk} be the event that the children $v_1, \ldots, v_{k-1}$ of v have no initial red nodes in their subtrees, node v_k has $\geq j$ initial red nodes in their subtrees, and children $v_{k+1}, \ldots, v_d$ of v have $\geq i-j$ initial red nodes in their subtrees. These events cover Q_2. Moreover, the probability of Q_{jk} is bounded by $(d+1) P_j(v) \cdot P_{i-j}(v)$: the event that exactly one of the nodes $v_{k+1}, \ldots, v_d$ has initial red descendants is covered by $d \cdot P_j(v) \cdot P_{i-j}(v)$. The event that more than one does has probability bounded by $P_{i-j}(v)$, which we multiply by the probability $P_j(v)$ that v_k has $\geq j$ descendants. Thus, in total, we get

$$Q_2 \leq 2d^2 \sum_{j=1}^{i-1} P_j(v) \cdot P_{i-j}(v) < 2d^2 \varepsilon_3 \cdot \frac{\varepsilon_3^{i+1}}{i^2} \cdot \sum_{j=1}^{i-1} \frac{i^2}{j^2 (i-j)^2} <$$

$$4d^2 \varepsilon_3 \cdot \frac{\varepsilon_3^{i+1}}{i^2} \cdot \sum_{j=1}^{\infty} \frac{4}{j^2} = \frac{8\pi^2}{3} d^2 \varepsilon_3 \cdot \frac{\varepsilon_3^{i+1}}{i^2} < \frac{\varepsilon_3^{i+1}}{i^2},$$

as long as $\varepsilon_3 < 3/(d^2 \cdot 8\pi^2)$. The claim follows immediately. ■

From now on we will assume that the conclusion of Claim 4.3 holds. Next we want to prove Lemma 4.2, namely that in each node the number of red blocks is bounded by α. We distinguish two types of red blocks: *natural* blocks and *extension* blocks. A red block is *natural* if either it is an initial red block, or the block is natural in one of the node's children. In other words, for each natural block in a node v there is a descendent of v connected to it via a red path where this block is an original red block. All other blocks are called *extension* blocks. Extension blocks occur because in the case when a node has two or more red children the process extends the red blocks by 1 before taking intersections.

We will bound the number of each type of blocks separately. As a first step, we present the process of red block creation above in an equivalent way as follows:

1. First of all, for each node in the tree we decide with probability ε_1 whether it contains any original red blocks at all; we also select for each node the random child that it copies;
2. we then sample the original blocks in the flagged "red" nodes conditioned on there being at least one red block;
3. we deterministically propagate the choices throughout the tree according to the rules.

Note that by Claim 4.3 the first step creates red connected components of size $< \varepsilon_2 \log n$. The propagation only happens on these connected components separately. Using this new presentation of the process we first bound the numbers of the natural red blocks in each node.

Claim 4.4 *Except with probability $< n^{-3}$ the maximum number of natural red blocks in each node is bounded by $\varepsilon_4 \log n$, where $\varepsilon_4 = 2\varepsilon_2$.*

Proof: We will prove that this is true for each individual node in the graph except with probability $n^{-3-\log d}$, thus implying the claim by union bound. Let v be a node and S_v be the nodes that contain at least one original red block, are in v's red connected component and that are v's descendants. By Claim 4.3 we know that $t = |S_v| < \varepsilon_2 \log n$. Denote the nodes in S_v by $v_1, \ldots, v_t$. Each node contains at least one original red block. Denote the number of red blocks in v_i, counted with multiplicities, by B_i. Then the B_i's are i.i.d. and for $j > 1$

$$\Pr[B_i > j] < \varepsilon_1^{j-1}.$$

since $\varepsilon_4 = 2\varepsilon_2$, and $\varepsilon_2 < \frac{3 \log d}{\log 1/\varepsilon_1}$, denoting $A = \varepsilon_4 \log n$, we have

$$\Pr\left[\sum_{j=1}^{t} B_j > A\right] = \sum_{i=A+1}^{\infty} \Pr\left[\sum_{j=1}^{t} B_j = i\right] < \sum_{i=A+1}^{\infty} \binom{i}{t} \varepsilon_1^{i-t} <$$

$$\sum_{i=A+1}^{\infty} 2^i \varepsilon_1^{i/2} < (4\varepsilon_1)^{A/2} < n^{-3-\log d}.$$

■

Next, we bound the number of extension red blocks. Note that extension blocks always have multiplicity 1. We again consider the original red blocks in each red connected component. Let S_v be a set of nodes that contain original red blocks and all belong to the same red connected component. We know that $|S_v| < \varepsilon_2 \log n$. We denote by P_k the set of blocks that are covered more than k times by original red blocks in S_v (not counting multiplicities). For example, P_1 is just the set of blocks that appear as original red blocks in at least one of the nodes of S_v. We first argue that

Claim 4.5 *For each k,*

$$\Pr[P_k > (\varepsilon_4 \log n)/k] < n^{-3-\log d}.$$

Thus, by union bound, we can assume that this even doesn't happen. The claim just follows from counting the total number of original red blocks. The proof of Claim 4.4 implies that the total number of original red blocks cannot exceed $\varepsilon_4 \log n$, and the claim follows by Markov's. Next we make a simple combinatorial observation.

Claim 4.6 *For each extension block b, there is a block b' that is i positions away from b such that $b' \in P_{2^{i/2-3}}$ (P_k is extended naturally to non-integer values by setting $P_k := P_{\lceil k \rceil}$).*

Proof: An extension block b can be traced to two children that are either in the original block or in an extension block as well. We can continue tracing the extension blocks until we obtain a binary tree with b at the root and an original red block at each leaf. Moreover, if the leaf is j levels from b then the location of its original block is $\leq j$-away from b. Denote the distances of all the leaf blocks from b by $d_1, \ldots, d_t$. We have

$$\sum_{j=1}^{t} 2^{-d_j} \geq 1.$$

Denote by n_k the number of leaf blocks *exactly* k-away from b (so that $\sum n_k = t$). Then we have

$$\sum_{k} n_k 2^{-k} \geq 1.$$

Algorithm 4: Trace reconstruction of a node x. Input: traces $\hat{x}_1, \ldots, \hat{x}_d$ of the children of x.

```
Let s ∈ [d] denote a random child ;
for each block k do
    G_k = {x̂_s[kB : kB + B]} ;
    h_s = 0 ;
    for each t ≠ s do
        if exists a shift −4α < h_t < 4α such that
        |x̂_s[kB : kB + B] ⊕ x̂_t[kB + h_t : kB + B + h_t]| < B/4 then
            Set G_k ← G_k ∪ {x̂_t[kB + h_t : kB + B + h_t]} ;
    Set x̂[kB : kB + B] = Majority_{σ∈G_k} σ
```

Hence there must exist a k such that $n_k > 2^{k/2}/4$. Otherwise

$$\sum_k n_k 2^{-k} < \frac{1}{4}\sum_k 2^{-k/2} = \frac{1}{4(1-1/\sqrt{2})} < 1.$$

Thus there is a location b' that is k-blocks away from b and that appears in at least $n_k/2 > 2^{k/2-3}$ original blocks, thus belonging to $P_{2^{k/2-3}}$. ■

Putting Claims 4.5 and 4.6 together we conclude that:

Claim 4.7 *Except with probability $< n^{-3}$ the total number of extension blocks in each connected component does not exceed $199\varepsilon_4 \log n$.*

Proof: Fix a connected component S_v. By Claim 4.6, each extension block is close to a point in one of the P_k's, and thus

$$\#\{\text{extension blocks in } S_v\} \leq \sum_{i=0}^{\infty}(2i+1)\cdot|P_{2^{i/2-3}}| \leq$$

by Claim 4.5

$$\varepsilon_4 \log n \cdot \sum_{i=0}^{\infty}(2i+1)\cdot 2^{3-i/2} < 199\varepsilon_4 \log n.$$

■

Lemma 4.2 is obtained by putting Claims 4.4 and 4.7 together. The former bounds the number of natural red blocks in every node, while the latter bounds the number of extension red blocks.

4.2 Trace Reconstruction Algorithm

Our trace reconstruction algorithm is presented in Alg. 4. We now explain the notation from the algorithm. α is the bound from Lemmas 4.1, 4.2, which satisfies $\alpha = O(\log n)$, $\alpha \ll B$. We will only care about the result of the majority, if at least $d-1$ children participated in it, treating it as adversarial otherwise. Note that since the original process is symmetric, the algorithm can easily be derandomized, fixing s to be, say, $s = 1$.

To analyse the algorithm, we introduce the following *trace reconstruction guarantee*, which serves as our inductive hypothesis. Let $\hat{x}_1, \ldots, \hat{x}_{\hat{K}}$ be the reconstructed sequence of a sequence x (trace of an internal node). We decompose it into consecutive blocks (subsequences) of length B, as we did in Section 4.1. Let $1 \leq R_1, \ldots, R_r \leq B$ denote the positions of the red blocks, where $r \leq \alpha$. Let n_i denote the number given to the i'th block if it's red, or $n_i = 0$ if i is a green block. Then there exists $g : [\hat{K}] \mapsto [-r, \ldots, r] \cup \{\perp\}$ satisfying the following properties:

1. $g(0) = 0$, by definition 0 is green.
2. g is not defined over red blocks: if $R_i B \leq j < R_i B + B$ for some $i \in [r]$, then $g(j) = \perp$.
3. g is constant over consecutive green blocks: If $j, j-1$ are both in green block(s), then $g(j) = g(j-1)$.
4. g can change by at most n_i over the i'th block: If j is green but $j-1$ is red, let $k < j$ be the last green site. Then
$$|g(j) - g(k)| \leq \sum_{k<i<j} n_i$$
where the sum ranges over all the red blocks between k and j.
5. g is an alignment of $\hat{x}$ and x: Consider the string Y which is aligning $\hat{x}$ according to g, that is $y[j] = \hat{x}[j + g(j)]$, or $y[j] = \perp$ if $g(j) = \perp$. Letting $z[j] = x[j] \oplus y[j]$, or 0 if $y[j] = \perp$, we have that z is stochastically dominated by a string which has 1 in each place with i.i.d probability β.

In words, for a site j in a green block, $g(j)$ will give the displacement between this site and the matching site in the original string x. g is not defined on sites in red blocks - it gives $\perp$. Note that although g is defined as a function of the site, it is actually equal for all the sites in the same block. The algorithm will not reconstruct g, but it will be used in the analysis.

We now prove that, if the reconstruction guarantee holds, then Alg. 4 performs the correct alignment.

Lemma 4.8 *Suppose that in children i and $j \neq s$ the k'th block is green. Then with high probability (that is with probability $1 - 2^{-\Omega(B)}$) there is exactly one shift $\hat{h}_j$ for which*

$$|\hat{x}_i[kB : kB + B] \oplus \hat{x}_j[\hat{h}_j + kB : kB + \hat{h}_j + B]| < (2\,\mathscr{P}_{\text{subs}}(1 - \mathscr{P}_{\text{subs}}) + 2\beta + \epsilon)\, B < B/4$$

This shift satisfies $-4\alpha < \hat{h}_j < 4\alpha$. Moreover, denoting $h_j = \hat{h}_j - g_j(kB) + g_i(kB)$, for each $1 \leq z \leq B$ we have $f_i^{-1}(kB + z) = f_j^{-1}(kB + z + h_j)$, or the same site in the father is mapped to the same site in the sons, up to the shift h_j.

Proof: The 4α bound comes from a maximal shift of 2α for each child. The 2α shift for each child is the sum of two terms: the maximal number of indel operations, and the bound on the shift $g_i(kB)$.

To show that the distance behaves correctly, note that

$$\hat{x}_i[kB + g_i(kB) : kB + g_i(kB) + B] \oplus x_i[kB : kB + B]$$

stochastically dominates a string which has 1 with i.i.d probability β. Since i, j are siblings, we know that there exists a shift h_j between them, such that

$$\Pr(x_i[kB + t] = x_j[kB + t + h_j]) = p_{s_i} p_{s_j} + (1 - p_{s_i})(1 - p_{s_j})$$

where $p_{s_i} < \mathscr{P}_{\text{subs}}$ is the substitution probability going from the father to child i, and $p_{s_j} < \mathscr{P}_{\text{subs}}$ is the substitution probability going from the father to child j. The shift h_j is just

$$h_j = |\{r < kB : f_i(r) = \perp\}| - |\{r < kB : f_j(r) = \perp\}| - |\{r < kB : f_i^{-1}(r) = \emptyset\}| + |\{r < kB : f_j^{-1}(r) = \emptyset\}|$$

or the difference between insertion and deletion operations between child i and child j up to block k. Note that we rely on the fact that there are no indel operations in blocks $k-1, k+1$ of the j'th child. This is the case, as otherwise block k would have been colored red, since $j \neq s$. Also note that the shift h_j may not be equal to $\hat{h}_j$, as h_j is the optimal shift in the ideal tree, and $\hat{h}_j$ is effected by the functions g_i, g_j.

Consider the reconstructed tree. The expected hamming distance of $\hat{x}_i[kB : kB + B]$ from $\hat{x}_j[kB + g_j(kB) - g_i(kB) + h_j : kB + g_j(kB) - g_i(kB) + h_j + B]$ is at most

$$\left(p_{s_i}(1 - p_{s_j}) + (1 - p_{s_i})p_{s_j} + 2\beta\right) B$$

, and we find $\hat{h}_j = h_j + g_j(kB) - g_i(kB)$.

Using a Chernoff bound and the definition of stochastic dominance, one can get that with high probability this distance is well concentrated. However, as for any $h \neq h_j$, we have

$$\Pr(x_i[kB + t] = x_j[kB + t + h]) = 1/2$$

And therefore the expected distance between $\hat{x}_i[kB : kB + B]$ and $\hat{x}_j[kB + g_j(kB) - g_i(kB) + h : kB + g_j(kB) - g_i(kB) + h + B]$ is at least $(1/2 - 2\beta)B - 2\alpha$, and again this distance is concentrated with high probability. As α is small relative to B, and β is a small enough constant, we get that with high probability the correct shift $\hat{h}_j$ passes the bound, and every other shift does not pass it. In this case, for each $1 \leq z \leq B$ we have $f_i^{-1}(kB+z) = f_j^{-1}(kB+z+h_j)$, where $h_j = \hat{h}_j - g_j(kB) + g_i(kB)$. ■

We condition on the high probability event of this lemma for any two comparisons between blocks made in the algorithm. This is a union bound over $\tilde{O}(n)$ comparisons, and the success probability of the lemma can be made to be $1 - 1/n^2$.

4.3 Existence of the Reconstruction Guarantee

We now show that the reconstruction guarantee holds inductively over the tree. We do so by showing that the algorithm alg. 4 is dominated by the random process analyzed in Section 4.1. We utilize Lemma 4.8 to argue that assuming the d children meet the reconstruction guarantee with some redness structure, the father also meets it, when the redness structure of the father is determined by the coloring procedure 3. Let $\vec{R}$ denote all the coin flips made by the algorithm; that is, R is a sequence of length $\sum_{i=0}^{\log n - 1} d^i$ of numbers in $[d]$, which choose which son was chosen in step 4 of Algorithm 3.

Lemma 4.9 *Suppose all the children meet the reconstruction guarantee for some red and green structure. Then the father x meets the guarantee as well, when the blocks of the father are colored red according to the coloring procedure 3, and the random choice of child node is made according to $\vec{R}$.*

Proof: Let g_s denote the alignment function between $\hat{x}_s$ and x_s, and let f_s denote the alignment function between the father node v to the child s. The set of sites which were deleted when going

from v to s is $D_s = \{j : f_s(j) = \perp\}$, and the set of sites which were inserted is $D_s = \{j : f_s^{-1}(j) = \emptyset\}$. We now define

$$g(j) = g_s(j) + |\{i \in D_s : \ i < j\}| - |\{i \in I_s : \ i < j\}|$$

As the sum of the numbers given to red blocks in the father before site j is at least $g_s(j) + D_s + I_s$, the definition satisfies condition 4 in the definition of g. As $g(j) \neq g(j-1)$ only when there is a red block in the father (either because $g_s(j) \neq g_s(j-1)$ or because there was an indel operation), g satisfies condition 3.

We now show that the reconstruction guarantee holds, given the alignment. Let k be a block which is green in s. Let G_k be the set of children for which k is green. If $|G_k| < d-1$, we make no claim about the result of the place-wise majority, as the coloring procedure 3 colors the k'th block red in the father. Otherwise applying lemma 4.8 between each one of the children in G_k and s, gives a set of shifts $\hat{H}_k$, such that for every $j \in S_k$ and site z

$$f_s^{-1}(kB + z) = f_j^{-1}(kB + z + \hat{h}_j) + g_s(kB) - g_j(kB)$$

Denote $h_j = \hat{h}_j - g_j(kB)$ and $a = f_s^{-1}(kB+z)$. Assume wlog that $x[a] = 1$. Let $b_j = x_j[kB+z+h_j]$ and $\hat{b}_j = \hat{x}_j[kB + z + \hat{h}_j]$ for $j \in G_k$, and let b be adversarial.

$$\begin{aligned} \Pr\left(\hat{x}[a + g(a)] = 1\right) &= \Pr\left(\sum_{j \in G_k} \hat{b}_j + b > d/2\right) \geq \Pr\left(\sum_{j \in G_k} \hat{b}_j > d/2\right) \quad (1) \\ &\geq \Pr\left(\sum_{j \in G_k} b_j > d/2 + 2\sqrt{d}\right) \Pr\left(|j : \ \hat{b}_j \neq b_j| < \sqrt{d}\right) \quad (2) \end{aligned}$$

However, $\sum_{j \in G} b_j$ is just a sum of indicator variables, with expectation

$$\mathbb{E} \sum_{j \in G} b_j \geq \sum_{j \in G} (1 - p_{s_j}) \geq \frac{d-1}{2} + (d-1)2\sqrt{\frac{\log d}{d}}$$

where p_{s_j} is the substitution probability going from the father to the j'th child, and $\mathscr{P}_{\text{subs}}$ is the bound on the substitution probability. Thus, we have $\Pr(\sum_{j \in G} b_j < d/2 + 2\sqrt{d}) < 1/2d^{2/3}$. As for the second term,

$$\Pr(|j : \ \hat{b}_j \neq b_j| > \sqrt{d}) = 2^{-O(d^{1/6})} < 1/2d^{2/3}$$

using $\beta = 1/d^{2/3}$ and d large enough. Putting this together gives that $\Pr(\hat{x}[j] = 1) > 1 - \beta$, and a similar analysis can be made when $x[j] = 0$.

Note that the event $\{\sum_{j \in G} b_j > d/2 + 2\sqrt{d}\} \bigcap \{|j : \ \hat{b}_j \neq b_j| < \sqrt{d}\}$ only depends on the i.i.d random variables which correspond to the substitutions, and on the sum of the random variables $a_j = \hat{b}_j \oplus b_j$, which are dominated by i.i.d random variables. Thus, if we let $y[j] = \hat{x}[j + g(j)] \oplus x[j]$, we have that y is stochastically dominated by a string which has 1 in each position with i.i.d probability β, as required. ■

4.4 Finding Siblings

In this section we finish the induction on levels, by showing that if all the nodes of level $\ell - 1$ match the reconstruction guarantee, then one can partition them to $d^{\log_d n - \ell}$ sets of size d, such that every

set will contain d siblings, or all the children of some node. We begin by defining a new distance, which is motivated by our reconstruction guarantee

$$\mathrm{d}_{\mathrm{ed}}(x, y, \gamma) = \min_{\mathrm{ed}_\gamma}(\mathrm{d}_{\mathrm{cor}}(\mathrm{ed}_\gamma(x), y))$$

Where $\mathrm{ed}_\gamma(x)$ is obtained from x by performing up to γ indel operations.

Claim 4.10 *There is an efficient algorithm which computes* $\mathrm{d}_{\mathrm{ed}}(x, y, \gamma)$.

The algorithm is based on dynamic programming.

Note that d_{ed} is not a metric, since it does not respect the triangle inequality in general. It is easy to see that the distance is monotone in γ, that is $\mathrm{d}_{\mathrm{ed}}(x, y, \gamma_1) \leq \mathrm{d}_{\mathrm{ed}}(x, y, \gamma_2)$ for $\gamma_1 \geq \gamma_2$. Moreover, when $\mathrm{d}_{\mathrm{ed}}(x, z, \gamma_1 + \gamma_2)$ is defined, the distance respects a limited form of triangle inequality

Claim 4.11

$$\mathrm{d}_{\mathrm{ed}}(x, y, \gamma_1) + \mathrm{d}_{\mathrm{ed}}(y, z, \gamma_2) \geq \mathrm{d}_{\mathrm{ed}}(x, z, \gamma_1 + \gamma_2)$$

The main tool that we want to use is neighbor joining (see e.g. [DMR06]). To use it, we need the following lemma. Let i, j be two nodes which are siblings, and v, w be arbitrary, such that all the pairwise distances are well defined. Then

Lemma 4.12 *With high probability,*

$$\mathrm{d}_{\mathrm{ed}}(\hat{x}_i, \hat{x}_j, 4\alpha B) + \mathrm{d}_{\mathrm{ed}}(\hat{x}_v, \hat{x}_w, 4\alpha B) < \mathrm{d}_{\mathrm{ed}}(\hat{x}_i, \hat{x}_v, 4\alpha B) + \mathrm{d}_{\mathrm{ed}}(\hat{x}_j, \hat{x}_w, 4\alpha B)$$

Note that if i, j are siblings then $\mathscr{A}(\hat{x}_i, \hat{x}_j, 4\alpha) > 0.75B^2$. Moreover, if they are not siblings there will be another vertex which will violate this equality (say the sibling of i).

Proof: We sketch the proof of this lemma. According to the triangle inequality, for any two vertices s, t

$$\mathrm{d}_{\mathrm{ed}}(\hat{x}_s, \hat{x}_t, 4\alpha B) \leq \mathrm{d}_{\mathrm{ed}}(\hat{x}_s, x_s, \alpha B) + \mathrm{d}_{\mathrm{ed}}(x_s, x_t, 2\alpha B) + \mathrm{d}_{\mathrm{ed}}(x_t, \hat{x}_t, \alpha B)$$

and similarly

$$\mathrm{d}_{\mathrm{ed}}(\hat{x}_s, \hat{x}_t, 4\alpha B) \geq \mathrm{d}_{\mathrm{ed}}(x_s, x_t, 6\alpha B) - \mathrm{d}_{\mathrm{ed}}(\hat{x}_s, x_s, \alpha B) - \mathrm{d}_{\mathrm{ed}}(x_t, \hat{x}_t, \alpha B)$$

The following claim is based on the reconstruction guarantee of $\hat{x}_v$

Claim 4.13 *With high probability,* $\mathscr{A}(\hat{x}_s, x_s, \alpha B) < 2\beta B^2 + \alpha B$

Proof: According to the reconstruction guarantee, implementing the alignment defined by the function g_s requires less than α edit operations. Given the alignment defined by g, the hamming distance between the cells in the green blocks of $\hat{x}_s$ and their counterparts in x_s is at most $2\beta B^2$, with exponentially good probability in B^2. Since there are at most α bad blocks, this can increase the distance by at most αB. ■

Let R_{st} denote the path on the tree from s to t, and let

$$p_{st} = \prod_{e \in R_{st}} (1 - 2p_e)$$

where p_e is the substitution probability of edge e, and we have $p_e \leq \mathscr{P}_{\mathrm{subs}}$.

Claim 4.14 *For any constant $\epsilon > 0$, with probability $2^{-\Omega(\epsilon^2 B)}$, we have $\mathscr{A}(x_s, x_t, 2\alpha B) < (1+\epsilon)\frac{2p_{st}+1}{2}B^2$.*

Proof: Let z be the common ancestor of s, t. According to Claim 4.1, with high probability there were at most αB indel operations on the path from z to s, and on the path from z to t. Conditioning on this event, both vertices can be aligned according to z. In this case, what we get is a simple hamming distance, which has exponentially good concentration. ■

We take $\epsilon = \beta$, which adds an error of the magnitude generated by Claim 4.13. We also need a lower bound on the distance, under the same condition $\mathscr{A}(x_s, x_t, 2\alpha B) > 0.75B^2$

Claim 4.15 *For any constant $\epsilon > 0$, with probability $2^{-\Omega(\epsilon^2 B)}$, we have $\mathscr{A}(x_s, x_t, 6\alpha B) > (1-\epsilon)\frac{p_{st}-1}{2}B^2$, as long as $\alpha \log(B/\alpha) \ll \varepsilon^2$.*

Proof: Fix an alignment of s, t. The probability that the distance is less than $(1-\epsilon)p_{st}B^2$ is at most $2^{-\Omega(\epsilon^2 B^2)}$, where the probability is taken over the substitutions, insertions and deletions of the random process which generated the tree. As there are at most

$$\binom{B^2}{6\alpha B} \leq (B/\alpha)^{6\alpha B} = 2^{6\alpha B \log(B/\alpha)} < 2^{O(\varepsilon^2 B^2)}$$

different alignment, it is possible to take a union bound for constant ϵ. ■

Again we take $\epsilon = \beta$.

Finally, Lemma 4.12 is proven by noticing that when $\beta, \alpha/B$ are small enough compared to the minimal substitution probability, and all the distances are small, we have that with high probability

$$\begin{aligned}
&\mathrm{d}_{\mathrm{ed}}(\hat{x}_i, \hat{x}_j, 4\alpha) + \mathrm{d}_{\mathrm{ed}}(\hat{x}_v, \hat{x}_w, 4\alpha) && (3)\\
&\quad \leq \log\left(\frac{2\,\mathscr{A}(\hat{x}_i, \hat{x}_j, 4\alpha)}{B^2} - 1\right) + \log\left(\frac{2\,\mathscr{A}(\hat{x}_v, \hat{x}_u, 4\alpha)}{B^2} - 1\right) && (4)\\
&\quad \leq \log(1+\beta)p_{ij}(1+4\beta)(1+2\alpha/B)(1+\beta)p_{vw}(1+4\beta)(1+2\alpha/B) && (5)\\
&\quad \leq \log(1-\beta)p_{iv}(1-4\beta)(1-2\alpha/B)(1-\beta)p_{jw}(1-4\beta)(1-2\alpha/B) && (6)\\
&\quad \leq \mathrm{d}_{\mathrm{ed}}(\hat{x}_i, \hat{x}_v, 4\alpha) + \mathrm{d}_{\mathrm{ed}}(\hat{x}_j, \hat{x}_w, 4\alpha) && (7)
\end{aligned}$$

where we substituted $\epsilon = \beta$, and used that

$$(1 - \mathscr{P}_{\min}) > \frac{(1+\beta)(1+4\beta)(1+2\alpha/B)(1+\beta)(1+4\beta)(1+2\alpha/B)}{(1-4\beta)(1-2\alpha/B)(1-\beta)(1-4\beta)(1-2\alpha/B)}$$

which holds as we choose α, β such that $\mathscr{P}_{\min} > 20\beta + 8\alpha/B$.

This complete proof of Lemma 4.12. ■

Given Lemma 4.12, it is straightforward to see the correctness of Algorithm 5. If a set S contains all the children of a single vertex, they will pass all tests. Otherwise, if S contains i, j which are not siblings, and v is a sibling of i, then according to Lemma 4.12, for every w, the test will fail, as $\mathrm{d}_{\mathrm{ed}}(\hat{x}_i, \hat{x}_j, 4\alpha) + \mathrm{d}_{\mathrm{ed}}(\hat{x}_v, \hat{x}_w, 4\alpha) < \mathrm{d}_{\mathrm{ed}}(\hat{x}_i, \hat{x}_v, 4\alpha) + \mathrm{d}_{\mathrm{ed}}(\hat{x}_j, \hat{x}_w, 4\alpha)$.

Note that in the current description we use the lemma $n^{O(d^2)}$ times. One can show that given the high probability event of Claim 4.1, and the reconstruction guarantee, the failure probability of Lemma 4.12 can be made $2^{-O(\log^2 n)}$, so this is not a problem. Also, it is easy to find more efficient algorithms which find siblings.

[10] The selection of the next cluster of degree d cousins can be done much more efficiently – in time $O(n^2)$. We omit the details to keep the presentation simpler.

Algorithm 5: Partition L, the nodes of level ℓ, into sets of siblings

```
for every[10] set S ⊂ L, with |S| = d do
    for every i, j ∈ S, and v, w ∈ L \ S do
        if d_ed(x̂_i, x̂_j, 4α) + d_ed(x̂_v, x̂_w, 4α) > d_ed(x̂_i, x̂_v, 4α) + d_ed(x̂_j, x̂_w, 4α) then
            S is not a set of siblings. Continue to the next set ;
        Add S to the partition
```

Algorithm 6: Reconstruction of a full binary tree with height $n \log d$. Let L_ℓ will be the set of vertices of level $\ell \log d$.

```
Let L_0 be the set of all vertices;
for level ℓ = log d, 2 log d, ... n log d do
    Initialize L_ℓ = ∅  for every[11] set S ⊂ L_{ℓ−1}, with |S| = d do
        for every i, j ∈ S, and v, w ∈ L \ S do
            if d_ed(x̂_i, x̂_j, 4α) + d_ed(x̂_v, x̂_w, 4α) > d_ed(x̂_i, x̂_v, 4α) + d_ed(x̂_j, x̂_w, 4α) then
                S is not a set of degree log d cousins. Continue to the next set ;
            L_ℓ ← L_ℓ ∪ {Reconstruct Binary Node(S)};
```

5 Phylogeny on Balanced Binary Trees

In this section, we present the algorithm for doing phylogenetic reconstruction on binary balanced trees, thereby proving Theorem 1.2.

5.1 Intuition and the reconstruction algorithm

We start by discussing the differences between the d-ary case and the binary one. For d a power of 2, let $Th(x_1, \ldots x_d)$ be defined as

$$Th(x_1, \ldots x_d) = \begin{cases} 1, & \sum x_i \geq 2d/3 \\ 0, & \sum x_i \leq d/3 \\ uniform \in \{0,1\}, & d/3 < \sum x_i < 2d/3 \end{cases}$$

The algorithm for reconstructing a complete binary tree is given in Figures 6, and 7. The algorithm is very similar to the one used for d-ary trees, with two differences:

1. The reconstruction procedure still requires d nodes. Thus, we reconstruct $\log d$ levels at a time, by taking degree $\log d$ cousins instead of just siblings.

2. When reconstructing a single node (Figure 7) we use the threshold function Th instead of using majority.

Intuitively, Algorithm 6 picks groups of d degree-$\log d$ siblings at a time, in a manner very similar to the degree-d case above. The main change is in Algorithm 6 — the reconstruction procedure for obtaining the sequence of a grandparent from its d degree-$\log d$ grandchildren. As before, we select a random grandchild and try to align the rest of the grandchildren to it. The major difference is

[11] Again, the selection of the next cluster of degree $\log d$ cousins can be done much more efficiently – in time $O(n^2)$.

Algorithm 7: Reconstruction of a single node in a binary tree. Inputs: $S = \{\hat{x}_1, \ldots, \hat{x}_d\}$

Let s denote a random descendent in S ;
for *each block* k **do**
  $G_k = \{\hat{x}_s[kB : kB + B]\}$;
  $h_s = 0$;
  for *each* $t \neq s$ **do**
    if *exists a shift* $-4\alpha < h_t < 4\alpha$ *such that* $|\hat{x}_s[kB : kB + B] \oplus \hat{x}_t[kB + h_t : kB + B + h_t]| < \frac{1}{4}B$ **then**
      Set $G_k \leftarrow G_k \cup \{\hat{x}_t[kB + h_t : kB + B + h_t]\}$;
  Set $\hat{x}[kB : kB + B] = Th_{B \in G_k} B$

in the application of the threshold function instead of a simple majority. If the k-th block is red in one of the grandchildren, we assume that this block is controlled by the adversary. As before we want to be able to tolerate the corruption of *one* of the grandchildren with high probability.

As explained before, the distance estimation algorithm may tolerate fairly high *random* reconstruction errors, since they are easily accounted for. Thus we have no problem tolerating a random reconstruction error β that is fairly high, say $\beta > \mathscr{P}_{\text{subs}}$. In fact β cannot be smaller than $\mathscr{P}_{\text{subs}}$, since if a mutation occurs on the edge to one of the immediate children of the grandparent we cannot expect to recover from the error. On the other hand, the distance estimation and neighbor joining algorithms are very sensitive to *adversarial* reconstruction errors. Here adversarial reconstruction errors are thought to occur *after* all the probabilistic choices have been made. We denote the adversarial error rate by γ. We want γ to be very small. In particular here we will show how to make $\gamma < \mathscr{P}_{\text{subs}}{}^3$.

Assuming only one (or a small constant number) of the grandchildren is controlled by the adversary, we want him to have very low control over the probability that the reconstruction is correct. Thus we want his expected influence to be bounded by γ. It is here that the second important modification plays a role. Instead of taking the majority over the leaves (i.e. threshold-(1/2)), we use a threshold-(1/3) function Th. Suppose we had used the threshold-(1/2) function. Assuming there is a mutation next to the root of the depth-$\log d$ tree (a probability-$\mathscr{P}_{\text{subs}}$ event) the number of leaves that disagree with the root is roughly $d/2$. Hence an adversary that controls even one leaf has a high chance of influencing the majority. Unlike 1/2, the number 1/3 does not have a finite binary representation. Hence if the number of leaves that disagree with the root is close to $d/3$ it means that many mutations must have occured (a low probability event!). Hence an adversary controlling just a small number of leaves is very unlikely to affect the outcome of Th, which is the cruicial property we need.

5.2 Correctness proof outline

The structure of the correctness proof is similar to the d-ary case. However, in the binary case the reconstruction guarantee is different. Let $g : [\hat{K}] \mapsto [-r, \ldots, r] \cup \{\perp\}$ be the alignment function between the reconstructed node $\hat{x}$ and the original node x. The behavior of g with respect to the red blocks is the same, but the behavior with respect to green blocks is different (property 5 of g in Section 4.2). Consider the string y which is aligning $\hat{x}$ according to g, that is $y[j] = \hat{x}[j + g(j)]$, or $y[j] = \perp$ if $g(j) = \perp$. Let $z[j] = x[j] \oplus y[j]$, or $\perp$ if $y[j] = \perp$. As in the d-ary case, we require that the part of z which does not contain $\perp$ is stochastically dominated by a string which has 1 in each place with i.i.d probability β. However, here we also require that the part of z which does

not involve the symbol $\perp$ stochastically dominates a string which has 1 in each place with i.i.d probability $\beta - \gamma$.

Formally, if the length of z is ℓ, there exists two random variables $z_{up}, z_{down} \in \{0, 1, \perp\}^\ell$ such that the joint distribution z_{down}, z, z_{up} satisfies the following properties:

1. The symbol $\perp$ occurs in the same places: if $z[j] = \perp$, also $z_{down}[j] = z_{up}[j] = \perp$, and if $z[j] \neq \perp$, also $z_{down}[j] \neq \perp$ and $z_{up}[j] \neq \perp$
2. If $z[j] = 0$ then $z_{down}[j] = 0$, and similarly if $z[j] = 1$ then $z_{up}[j] = 1$.
3. The marginal distribution on $z_{down} \times z_{up}$ where $z[j] \neq \perp$ is i.i.d such that $z_{up}[j] = 1$ with probability β; $z_{up}[j] = 1$ whenever $z_{down}[j] = 1$, and $\mathbf{P}[z_{up}[j] = 1 | z_{down}[j] = 0] = \gamma$.

Throughout the construction we will maintain the invariant that β is small. More specifically, we will have

$$\beta < {\mathscr{P}_{\text{subs}}}^{2/3}.$$

We will show that γ can be maintained very small. Specifically, we will show how to maintain

$$\gamma < {\mathscr{P}_{\text{subs}}}^{3}.$$

It should be noted that for a sufficiently small constant $\mathscr{P}_{\text{subs}}$ a bound $\gamma < {\mathscr{P}_{\text{subs}}}^{c}$ can be realized with any constant $c > 1$.

Given this guarantee, there are three differences between the binary case and the d-ary case, which will be described in the following sections.

5.3 Red and green trees

Recall that the red and green trees were introduced to control the (red) locations that have been affected by indels making matching them to their cousins by a shift of magnitude α potentially impossible. Since the reconstruction is done in batches of d nodes, we need to modify the algorithm that controls errors (Algorithm 3) slightly to obtain Algorithm 8. In Algorithm 3 indels that occurred going from t's parent to t in block k caused block k in t to become red. In Algorithm 8 we charge indels that occur between a node t and its depth-$\log d$ grandchildren to t. We do this since the number of indels occuring between t and two of its descendants t_1 and t_2 are no longer independent. For example, if an indel occurs between t and its immediate child, with high probability the same indel will occur between t and half of its descendants.

The analysis of Algorithm 8 is very similar to the analysis of Algorithm 3. The only difference is that in the initialization the $n_{k,t}$ are generally larger in Algorithm 8. However, is easy to see that the process in Algorithm 8 is dominated by the process in Algorithm 3 with $\mathscr{P}_{\text{id}}$ replaced with $2d \cdot \mathscr{P}_{\text{id}}$. Thus we get the following analogue of Lemma 4.2:

Lemma 5.1 *With high probability, in each node of the tree there are at most α blocks which are not green, assuming $\alpha \geq \frac{1200 \log d}{\log 1/(2B^2 \cdot d \cdot \mathscr{P}_{\text{id}})}$.*

5.4 Distance estimation

Given this guarantee, we need to prove an analog of Lemma 4.12. That is, when i, j are siblings and the distances are well defined then

Algorithm 8: Recursively coloring a depth-$\log d$ grandfather y given the d grandchild nodes

Initialization:
for *each vertex* t*, and each block* k **do**
  if *the maximum number of insertions and deletions going from* t *to any of its* depth-$\log d$ grandchildren *in block* k *is* $n_{k,t} > 0$ **then**
    Color block k in t red, and give it the number $n_{k,t} > 0$;
Recursively coloring a grandfather y **given the** d **grandchild nodes:**
Let s be a random grandchild ;
for *every child* $t \neq s$ **do**
  for *every maximal consecutive sequence of red blocks in the* t*'th grandchild,* $i, i+1, \ldots, i+k$ **do**
    temporarily color blocks $i-1, i+k+1$ red in the grandchild t, and give them the number 1.
for $k = 1$ *to* B **do**
  if *the* k*'th block in* s *is red, and has number* i **then**
    Color the k'th block in the grandfather red, and add number i to it.
  else
    if *exist* $t_1 \neq t_2$ *in which the* k*'th block is red* **then**
      Color the k'th block in the grandfather red, and give it number 1.

Lemma 5.2 *With high probability,*

$$\mathrm{d_{ed}}(\hat{x}_i, \hat{x}_j, 4\alpha) + \mathrm{d_{ed}}(\hat{x}_v, \hat{x}_w, 4\alpha) < \mathrm{d_{ed}}(\hat{x}_i, \hat{x}_v, 4\alpha) + \mathrm{d_{ed}}(\hat{x}_j, \hat{x}_w, 4\alpha)$$

Proof:(Sketch) The lemma has two main parts:

1. Showing that the edit distance operations and the red blocks do not change the distance by much. This is done using the triangle inequality (Claim 4.11)
2. Showing that the distance guarantee on the good blocks is good enough.

The first part is very similar to the proof of Lemma 4.12. As for the second part, by choosing $\epsilon = \gamma$ in claims 4.14, 4.15, we get that the equation 3 changes to

$$\begin{aligned}
\mathrm{d_{ed}}(\hat{x}_i, \hat{x}_j, 4\alpha) + \mathrm{d_{ed}}(\hat{x}_v, \hat{x}_w, 4\alpha) &\leq \log\left(\frac{2\,\mathscr{A}(\hat{x}_i, \hat{x}_j, 4\alpha)}{B^2} - 1\right) + \log\left(\frac{2\,\mathscr{A}(\hat{x}_v, \hat{x}_u, 4\alpha)}{B^2} - 1\right) \\
&\leq \log(1-\beta_i)(1-\beta_j)p_{ij}(1+2\alpha/B)(1+2\gamma)(1-\beta_v)(1-\beta_w)p_{vw}(1+2\gamma)(1+2\alpha/B)(1+\gamma) \\
&\leq \log(1-\beta_i)(1-\beta_v)p_{iv}(1-2\alpha/B)(1-2\gamma)(1-\beta_j)(1-\beta_w)p_{jw}(1-2\alpha/B)(1-2\gamma)(1-\gamma) \\
&\leq \mathrm{d_{ed}}(\hat{x}_i, \hat{x}_v, 4\alpha) + \mathrm{d_{ed}}(\hat{x}_j, \hat{x}_w, 4\alpha)
\end{aligned}$$

and this holds given that

$$1 - \mathscr{P}_{\min} \geq \frac{(1-\gamma)(1-2\gamma)^2(1-2\alpha/B)^2}{(1+\gamma)(1+2\gamma)^2(1+2\alpha/B)^2}$$

which is again true if $\mathscr{P}_{\min} > 4\alpha/B + 10\gamma$. ■

5.5 Meeting the recursive guarantee

Finally, we need to show that if the grandchildren meet the reconstruction guarantee, so does the grandfather. In the d-ary case, this was done in Lemma 4.9. In the binary case, the behavior of red blocks is similar, but we need an estimate on the adversary's influence, which replaces (1) in the proof of the lemma. It is here that the main difference between the d-ary and the binary case occurs. The reconstruction process as taking grandchildren nodes with $z^1, \ldots, z^d \in \{0, 1, \perp\}^\ell$ representing their reconstruction errors and uses Algorithm 7 to generate the grandparent node with z representing its reconstruction errors. By the recursive guarantee, it is useful to think of the z^i's as representing two types of errors: a fairly large random noise $\beta_i < \mathscr{P}_{\text{subs}}{}^{2/3}$ as represented by z^i_{up}, and a small adversarial error may be subtracted with probability $\leq \gamma$. The β_i's may depend on the edge lengths and may vary among the z^i's. On the other hand, we assume that the same $\gamma \ll \beta_i$ is fixed throughout the entire reconstruction. Since the stochastic domination is i.i.d on all the locations $j = 1, .., \ell$, it is enough to show that the recursive guarantee is preserved location-wise. From now on, we fix a location j. We need to prove the following recursive guarantee:

- If $z^i[j] = \perp$ for at most one $i \in \{1, .., d\}$, then there is a $\beta < \mathscr{P}_{\text{subs}}{}^{2/3}$ such that $z[j]$ stochastically dominates $z_{down}[j]$ and is dominated by $z_{up}[j]$ where $\Pr[z_{up}[j] = 1] = \beta$ and $\Pr[z_{up} = 1 | z_{down}[j] = 0] = \gamma$.

Thus the adversary is given full control over one of the d grandchildren (the one where $z^i[j] = \perp$), and is given control over each of the other grandchildren with probability $< \gamma$, our goal is to show that if we apply the procedure from Algorithm 7 the probability of the adversary gaining control over the root value $z[j]$ is tiny ($< \gamma$).

Consider the process where each $z^i[j] = z^i_{down}[j]$ is sampled independently to be 1 with probability β_i, and then the mutations $m^1[j], \ldots, m^d[j]$ in the phylogenic tree on location j between $z^1, \ldots, z^d$ and z are sampled (possibly with the adversary's interference). The reconstruction works correctly with probability

$$p_0 := \Pr\left[\sum_{i=1}^{d} z^i[j] \oplus m^i[j] \leq \frac{d}{3}\right] + \frac{1}{2}\Pr\left[\frac{d}{3} < \sum_{i=1}^{d} z^i[j] \oplus m^i[j] < \frac{2d}{3}\right].$$

The first term accounts for the case when the total number of disagreements between $z[j]$ and its (reconstructed) descendants is at most $d/3$, and the second term accounts for the case when this number is between $d/3$ and $2d/3$. For simplicity assume that $1/\sqrt{\mathscr{P}_{\text{subs}}}$ is a power of 2. Set

$$d := 1/\sqrt{\mathscr{P}_{\text{subs}}}.$$

We have $d^2\gamma^2 < \gamma/4$, and hence the probability that the adversary gains control over two of the $z^i[j]$'s is bounded by $\gamma/4$.

It is not hard to see that assuming that at most one $z^i[j] = \perp$,

Claim 5.3

$$\Pr[\#\{i\text{'s such that } z^i[j] \neq 0\} \geq 20] < \gamma/4.$$

Thus all but less than 20 grandchild nodes are reconstructed correctly. This estimate includes potential interventions by the adversary who may control 2 of the $z^i[j]$'s. The number of edges between z and the grandchildren is $2d - 2 \ll 1/\mathscr{P}_{\text{subs}}$, and thus with high probability the number of mutations on these nodes is very small:

Claim 5.4

$$\Pr[\#\textit{ of mutations in the subtree} \geq 7] < \gamma/4.$$

There are 30 edges in the first four layers of the tree. Hence the probability of having a mutation in one of these edges is bounded by

$$30 \cdot \mathscr{P}_{\text{subs}} < {\mathscr{P}_{\text{subs}}}^{2/3}/2,$$

for a sufficiently small constant $\mathscr{P}_{\text{subs}}$. If there are no mutations in the first four levels, and assuming the conclusion of Claim 5.4, we have

$$\sum_{i=1}^{d} m^i[j] \leq 2^{-5} d \cdot 7 = \frac{7d}{32}.$$

Putting this together with Claim 5.3, we see that except with probability $< {\mathscr{P}_{\text{subs}}}^{2/3}/2 + \gamma/2 < {\mathscr{P}_{\text{subs}}}^{2/3}$,

$$\sum_{i=1}^{d} z^i[j] \oplus m^i[j] \leq \sum_{i=1}^{d} m^i[j] + \sum_{i=1}^{d} z^i[j] < \frac{7d}{32} + 20 < \frac{d}{3}$$

for a small constant $\mathscr{P}_{\text{subs}}$. Thus the event of the grandparent value being reconstructed incorrectly is bounded by a probability $\beta < {\mathscr{P}_{\text{subs}}}^{2/3}$.

The most important part is estimating the probability γ' that the adversary can manipulate the reconstruction output, and showing that it is bounded by γ. Since the adversary controls at most two inputs, the probability that he will be able to change the output is bounded by

$$\Pr\left[\sum_{i=1}^{d} z^i[j] \oplus m^i[j] \in (d/3-2, d/3+2) \cup (2d/3-2, 2d/3+2)\right].$$

We will show that assuming the conclusions of Claims 5.3 and 5.4

$$\sum_{i=1}^{d} z^i[j] \oplus m^i[j] \notin (d/3-2, d/3+2) \cup (2d/3-2, 2d/3+2), \tag{8}$$

and thus $\gamma' < \gamma/2 < \gamma$.

Define the following subset of the unit interval:

$$S := \{x \in [0,1] \mid x = \sum_{i=1}^{7} \eta_i 2^{-t_i}, \text{ where } \eta_i = \pm 1 \text{ and } t_i > 0 \text{ is an integer}\}.$$

Thus S is the set of numbers that have a representation using a signed sum of at most 7 inverse powers of 2. It is not hard to see that there is a constant $\tau > 0$ such that $|x - 1/3| > \tau$ and $|x - 2/3| > \tau$ for all $x \in S$. We select $\mathscr{P}_{\text{subs}}$ sufficiently small, so that $d = 1/\sqrt{\mathscr{P}_{\text{subs}}} > 23/\tau$.

Observe that assuming the conclusion of Claim 5.4 we have

$$\sum_{i=1}^{d} m^i[j] = x \cdot d,$$

where $x \in S$, and assuming Claim 5.3 we have

$$\left|\sum_{i=1}^{d} z^i[j] \oplus m^i[j] - d/3\right| \geq \left|\sum_{i=1}^{d} m^i[j] - d/3\right| + \sum_{i=1}^{d} z^i[j] > \tau \cdot d - 20 > 3,$$

and hence

$$\sum_{i=1}^{d} z^i[j] \oplus m^i[j] \notin (d/3 - 2, d/3 + 2).$$

A similar argument with $2d/3$ instead of $d/3$ concludes the proof of (8).

It should be noted that the use of the $d/3$ threshold instead of the more "natural" $d/2$ threshold for the majority vote is crucial in our analysis. With probability as high as $\mathscr{P}_{\text{subs}}$ one of the edges adjacent to the root contains a mutation, and hence about one half of the $m^i[j]$'s are 1. Hence an adversary controlling even a single leaf may influence the value of the majority. On the other hand, since $1/3$ does not have a "nice" binary representation, a sequence of mutations such that about one third of the $m^i[j]$'s are 1 is unlikely. In the proof we used the fact that $1/3$ is removed from the set S. At the same time $1/2$ is contained in the set S, making $\tau = 0$.

6 Phylogeny on General Binary Trees

In this section, we show how our results generalize to binary trees with arbitrary topologies, in particular, to trees without a restriction on the depth, thereby proving Theorem 1.3. To obtain this result, we rely on the algorithm of [MHR08], that performs phylogeny reconstruction in the standard CFN model (where there are no insertions and deletions). In particular, we show that we can use the algorithm from [MHR08] in an essentially black-box fashion.

We give a brief overview of the [MHR08] algorithm. Their algorithm constructs the tree iteratively, maintaining a forest induced from T. At each step, the algorithm joins two trees of the forest into a single tree. In the process, they create internal nodes, for which one computes a "reconstructed trace" from the nodes in same tree. To connect the trees, the algorithm then uses empirical evaluations of the distance between traces of the nodes. The traces are reconstructed via now-classical recursive majority procedure of [Mos98, Mos01, Mos03, Mos04b].

We use the [MHR08] algorithm with two modifications. The first one is that we use our trace reconstruction (computation of learned characters) from Algorithm 7 instead of the recursive majority rule used in MHR. The second modification is to replace their distance estimation with our distance estimation, d_{ed}, and applying the trace reconstruction procedure from previous sections.

We now state the guarantees that are achieved by (a modified version of) [MHR08] algorithm. In particular, we consider the MHR algorithm, with trace reconstruction that uses threshold function Th instead of majority Maj (in step (4)(h)(ii) in TREE-MERGE algorithm). Call this algorithm MHR-TH.

We also need the notion of an (M, ε)-*approximator*: a quantity $\hat{D}$ is a (M, ε)-approximator of D if: $|\hat{D} - D| \leq \varepsilon$ when $D < M$ and $\hat{D} > M - \varepsilon$ when $D \geq M$. Also, for two random variables $x, y \in \{+1, -1\}$, the MHR-distance is[12]

$$\mathrm{d}_{\mathrm{Pr}}(x, y) = -\log\left(2\Pr[x = y] - 1\right).$$

Theorem 6.1 ([MHR08], see Theorem 4.5) *Consider an arbitrary binary tree T on n leafs, with a reversible CFN model on k sites. Suppose the substitution probability of each edge is $12\varepsilon \leq p_e \leq \lambda - 6\varepsilon$, for some $\varepsilon > 0$, where $\lambda \leq \lambda_0$ for λ_0 being the phase transition of the CFN model.*

Furthermore, suppose we have query access to a $(M, \varepsilon/4)$-approximator of the d_{Pr} distance, for $M = c_M\lambda + c_M\varepsilon$, for some absolute constant $c_M > 1$.

Then the MHR-TH algorithm reconstructs the tree T from the observed sites on the leafs, with high probability. The algorithm runs in polynomial time.

[12] In [MHR08], there is also a normalization factor of 0.5, which of course does not affect the algorithm.

Now we show how to obtain theorem 1.3. We use the following algorithm, termed MHR-INDEL. It is the algorithm from [MHR08], where the trace reconstruction (step (4)(h)(ii) of TREE-MERGE) is performed using Algorithm 7. Also all distances are computed using $\mathrm{d}_{\mathrm{ed}}(\cdot,\cdot,4\alpha \cdot M/\mathscr{P}_{\min})$ (for both leafs and reconstructed traces of internal nodes).

To prove the correctness guarantee of algorithm MHR-INDEL, we use Theorem 6.1 with the following parameters $\varepsilon = \mathscr{P}_{\min}/12$, and $\lambda = \mathscr{P}_{\mathrm{subs}} + 6\varepsilon \leq \lambda_0$.

We first introduce an intermediary algorithm, termed MHR-IDEAL-INDEL. Consider the same mutation process INDEL-REV. Now suppose all trace reconstructions are ideal: for a node v that is reconstructed from a subtree T, a site j from v is reconstructed from corresponding sites in the nodes in T, using the thresholding function Th. (If not all nodes in T contain the site j — a situation possible due to insertions/deletions — then use the nodes where the site is available.) We also define the following ideal distance $\mathrm{d}_{\mathrm{indel}}$, between two strings $x, y \in \{0,1\}^k$, which we assume share l sites S_x in x and S_y in y, given by the bijective map $m : S_x \to S_y$:

$$\mathrm{d}_{\mathrm{indel}} = \frac{1}{l}\left(-\sum_{i \in S_x} \log(2\Pr[x_i = y_{m(i)}] - 1) + (k-l)M\right).$$

In other words, the $\mathrm{d}_{\mathrm{indel}}$ is just the average of the standard d_{Pr} distance over all sites, except that for unshared sites, we use the max distance M.

We claim that the MHR-IDEAL-INDEL (i.e., under the mutation process INDEL-REV, and $\mathrm{d}_{\mathrm{indel}}$ distance) succeeds in recovering the phylogeny tree T with high probability. Indeed, first we notice that all independence conditions in MHR hold true (for each site separately, and hence for the entire sequences). Second, consider any two sequences x, y. If they are at distance $\mathrm{d}_{\mathrm{Pr}} \geq M - \varepsilon$ in the MHR-TH algorithm, then they are at distance $\mathrm{d}_{\mathrm{indel}} \geq M - \varepsilon$ in the MHR-IDEAL-IDEAL algorithm as well (since this holds for each site separately and $\mathrm{d}_{\mathrm{indel}}$ is just an average). Otherwise, x and y belong to nodes that are at distance at most $M/\mathscr{P}_{\min} \leq 2c_M \cdot \mathscr{P}_{\mathrm{subs}}/\mathscr{P}_{\min}$. Consider all the nodes C, at distance $2M$ from x (note that $y \in C$). Then, with high probability (over indel mutations), $k - O(\log n) = k(1 - o(1))$ sites of x are present in each node in C. Each site behaves like a standard CNF model, with Threshold reconstruction. Hence, if we restrict to the sites shared by all of C, then, the algorithm that would use these sites in the computation of $\mathrm{d}_{\mathrm{indel}}$ would behave exactly the same as MHR-TH. Now the actual $\mathrm{d}_{\mathrm{indel}}$ involves more sites, not all shared inside C, but the effect on the distance is only $1 - o(1)$, and hence it is a $(M, \varepsilon/4)$-approximator to the previous distance.

Finally, we observe that the distance computations in our actual algorithm MHR-INDEL are good approximators to the distances in the above idealized MHR-IDEAL-INDEL, with high probability (over indel and substitution mutations). Indeed, in Section 5, we showed the recursive guarantee for each reconstructed trace. The reconstruction guarantee says that each trace is reconstructed according to the thresholding function Th, except that a fraction of sites fall pray to adversary: in fact, at most $\alpha B + \gamma k$ sites are controlled by adversary (these includes the sites where $z[j] = \perp$ and $z[j]$ is controlled by the adversary with probability γ). The number of such sites is $\alpha B + \gamma k \ll \varepsilon/8 \cdot k$. For the rest of the sites, which are $\geq k \cdot (1 - \varepsilon/8)$, we can apply a standard concentration bound, concluding that the empirical distance $\mathrm{d}_{\mathrm{ed}}(\cdot,\cdot,4\alpha \cdot M/\mathscr{P}_{\min})$ is a $(M, \varepsilon/4)$-approximator to $\mathrm{d}_{\mathrm{indel}}(\cdot,\cdot)$.

7 Acknowledgments

We are grateful to Constantinos Daskalakis and Sébastien Roch for many helpful discussions.

References

[ADHR10] Alex Andoni, Constantinos Daskalakis, Avinatan Hassidim, and Sébastien Roch. Trace reconstruction on a tree. In *ICS*, 2010.

[BCMR06] Christian Borgs, Jennifer T. Chayes, Elchanan Mossel, and Sébastien Roch. The Kesten-Stigum reconstruction bound is tight for roughly symmetric binary channels. In *FOCS*, pages 518–530, 2006.

[BKMP05] N. Berger, C. Kenyon, E. Mossel, and Y. Peres. Glauber dynamics on trees and hyperbolic graphs. *Probab. Theory Rel.*, 131(3):311–340, 2005. Extended abstract by Kenyon, Mossel and Peres appeared in proceedings of 42nd IEEE Symposium on Foundations of Computer Science (FOCS) 2001, 568–578.

[BRZ95] P. M. Bleher, J. Ruiz, and V. A. Zagrebnov. On the purity of the limiting Gibbs state for the Ising model on the Bethe lattice. *J. Statist. Phys.*, 79(1-2):473–482, 1995.

[Cav78] J. A. Cavender. Taxonomy with confidence. *Math. Biosci.*, 40(3-4), 1978.

[DMR06] Constantinos Daskalakis, Elchanan Mossel, and Sébastien Roch. Optimal phylogenetic reconstruction. In *STOC'06: Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 159–168, New York, 2006. ACM.

[DR10] Constantinos Daskalakis and Sébastien Roch. Alignment-free phylogenetic reconstruction. In *RECOMB*, 2010.

[EKPS00] W. S. Evans, C. Kenyon, Y. Peres, and L. J. Schulman. Broadcasting on trees and the Ising model. *Ann. Appl. Probab.*, 10(2):410–433, 2000.

[ESSW99a] P. L. Erdös, M. A. Steel, L. A. Székely, and T. A. Warnow. A few logs suffice to build (almost) all trees (part 1). *Random Struct. Algor.*, 14(2):153–184, 1999.

[ESSW99b] P. L. Erdös, M. A. Steel, L. A. Székely, and T. A. Warnow. A few logs suffice to build (almost) all trees (part 2). *Theor. Comput. Sci.*, 221:77–118, 1999.

[Far73] J. S. Farris. A probability model for inferring evolutionary trees. *Syst. Zool.*, 22(4):250–256, 1973.

[Fel04] J. Felsenstein. *Inferring Phylogenies*. Sinauer, New York, New York, 2004.

[Iof96] D. Ioffe. On the extremality of the disordered state for the Ising model on the Bethe lattice. *Lett. Math. Phys.*, 37(2):137–143, 1996.

[LG08] Ari Loytynoja and Nick Goldman. Phylogeny-Aware Gap Placement Prevents Errors in Sequence Alignment and Evolutionary Analysis. *Science*, 320(5883):1632–1635, 2008.

[LRN+09a] K. Liu, S. Raghavan, S. Nelesen, C.R. Linder, and T. Warnow. Rapid and accurate large-scale coestimation of sequence alignments and phylogenetic trees. *Science*, 324(5934):1561, 2009.

[LRN+09b] Kevin Liu, Sindhu Raghavan, Serita Nelesen, C. Randal Linder, and Tandy Warnow. Rapid and Accurate Large-Scale Coestimation of Sequence Alignments and Phylogenetic Trees. *Science*, 324(5934):1561–1564, 2009.

[Met03] Dirk Metzler. Statistical alignment based on fragment insertion and deletion models. *Bioinformatics*, 19(4):490–499, 2003.

[MHR08] Radu Mihaescu, Cameron Hill, and Satish Rao. Fast phylogeny reconstruction through learning of ancestral sequences. *CoRR*, abs/0812.1587, 2008.

[MLH04] I. Miklos, G. A. Lunter, and I. Holmes. A "Long Indel" Model For Evolutionary Sequence Alignment. *Mol Biol Evol*, 21(3):529–540, 2004.

[Mos98] E. Mossel. Recursive reconstruction on periodic trees. *Random Struct. Algor.*, 13(1):81–97, 1998.

[Mos01] E. Mossel. Reconstruction on trees: beating the second eigenvalue. *Ann. Appl. Probab.*, 11(1):285–300, 2001.

[Mos03] E. Mossel. On the impossibility of reconstructing ancestral data and phylogenies. *J. Comput. Biol.*, 10(5):669–678, 2003.

[Mos04a] E. Mossel. Phase transitions in phylogeny. *Trans. Amer. Math. Soc.*, 356(6):2379–2404, 2004.

[Mos04b] E. Mossel. Survey: Information flow on trees. In J. Nestril and P. Winkler, editors, *Graphs, morphisms and statistical physics*, pages 155–170. Amer. Math. Soc., 2004.

[MSW04] F. Martinelli, A. Sinclair, and D. Weitz. Glauber dynamics on trees: boundary conditions and mixing time. *Comm. Math. Phys.*, 250(2):301–334, 2004.

[Ney71] J. Neyman. Molecular studies of evolution: a source of novel statistical problems. In S. S. Gupta and J. Yackel, editors, *Statistical desicion theory and related topics*, pages 1–27. Academic Press, New York, 1971.

[RE08] Elena Rivas and Sean R. Eddy. Probabilistic phylogenetic inference with insertions and deletions. *PLoS Comput Biol*, 4(9):e1000172, 09 2008.

[Roc07] S. Roch. *Markov Models on Trees: Reconstruction and Applications*. PhD thesis, UC Berkeley, 2007.

[Roc08] Sébastien Roch. Sequence-length requirement for distance-based phylogeny reconstruction: Breaking the polynomial barrier. In *FOCS*, pages 729–738, 2008.

[SR06] Marc A. Suchard and Benjamin D. Redelings. BAli-Phy: simultaneous Bayesian inference of alignment and phylogeny. *Bioinformatics*, 22(16):2047–2048, 2006.

[SS03] C. Semple and M. Steel. *Phylogenetics*, volume 22 of *Mathematics and its Applications series*. Oxford University Press, 2003.

[TKF91] Jeffrey L. Thorne, Hirohisa Kishino, and Joseph Felsenstein. An evolutionary model for maximum likelihood alignment of dna sequences. *Journal of Molecular Evolution*, 33(2):114–124, 1991.

[TKF92] Jeffrey L. Thorne, Hirohisa Kishino, and Joseph Felsenstein. Inching toward reality: An improved likelihood model of sequence evolution. *Journal of Molecular Evolution*, 34(1):3–16, 1992.

[WSH08] Karen M. Wong, Marc A. Suchard, and John P. Huelsenbeck. Alignment Uncertainty and Genomic Analysis. *Science*, 319(5862):473–476, 2008.